

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MediaTek Inc.,

Petitioner

v.

Daedalus Prime LLC,

Patent Owner

Case: IPR2024-01488

U.S. Patent No. 10,740,281

Issue Date: August 11, 2020

Title: Asymmetric Performance Multicore Architecture with Same
Instruction Set Architecture

**DECLARATION OF MURALI ANNAVARAM PH.D.
IN SUPPORT OF PETITION FOR INTER PARTES REVIEW OF
U.S. PATENT NO. 10,740,281**

TABLE OF CONTENTS

I.	Introduction.....	1
II.	Qualifications and Background	1
III.	Materials Considered.....	7
IV.	Legal Standards	8
	A. Claim Construction	9
	B. Obviousness Under 35 U.S.C. § 103.....	10
	C. Secondary Considerations of Non-Obviousness.....	12
V.	The '281 Patent.....	14
	A. Overview of the '281 Patent.....	14
	B. Summary of the Prosecution History of the '281 Patent	15
	C. Claim Construction of Terms in the '281 Patent	16
VI.	Person of Ordinary Skill in the Art.....	16
VII.	The Prior Art.....	18
	A. Overview of Sutardja (EX-1002)	18
	B. Overview of Mathieson (EX-1003).....	19
	C. Overview of Patterson-1 (EX-1004)	20
	D. Overview of Knebel (EX-1005).....	21
	E. Overview of Liu (EX-1006).....	21
	F. Overview of Sakarda (EX-1007).....	22
VIII.	Opinions on Unpatentability.....	23
	A. Ground 1A: Sutardja (EX-1002) and Mathieson (EX-1003) renders obvious claims 1, 2, 4, 6, 8, 9, 11, 13, 15, 16, 18, and 20.....	23

1.	Motivation to Combine Sutardja and Mathieson.....	23
2.	Claim 1	28
3.	Claims 2, 9, 16: “wherein each of the first plurality of cores and each of the second plurality of cores further include a respective caching layer therein.”	45
4.	Claims 4, 11, 18: “wherein the core mix includes a state where some of, but not all of, the first plurality of cores and the second plurality of cores are enabled.”	47
5.	Claims 6, 13, 20: “wherein the first plurality of cores is at least three.”	48
6.	Claim 8	50
7.	Claim 15	53
B.	Ground 1B: Sutardja (EX-1002), Mathieson (EX-1003), and Patterson-1 (EX-1004) renders obvious claims 3, 10, and 17	54
1.	Claims 3, 10, 17: “wherein each of the first plurality of cores includes therein a caching layer for data and a caching layer for instructions.”	54
C.	Ground 1C: Sutardja (EX-1002), Mathieson (EX-1003), and Knebel (EX-1005) renders obvious claims 5, 12, and 19	58
1.	Claim 5, 12, 19: “a system memory interface to couple to a system memory to search for a cache line, requested by any core of the first plurality of cores and the second plurality of cores, that is not found in the caching layer” / “coupling a system memory interface of the multi-core processor to a system memory to search for a cache line, requested by any core of the first plurality of cores and the second plurality of cores, that is not found in the caching layer”	58
D.	Ground 1D: Sutardja (EX-1002), Mathieson (EX-1003), and Liu (EX-1006) renders obvious claims 7, 14, and 21	63
1.	Claims 7, 14, 21: “coherency logic circuitry”	63

E.	Ground 2A: Mathieson (EX-1003) and Sakarda (EX-1007) renders obvious claims 1-4, 6, 8-11, 13, 15-18, and 20	66
1.	Motivation to Combine Mathieson and Sakarda	66
	First Mathieson-Sakarda Combination	66
	Second Mathieson-Sakarda Combination	69
2.	Claim 1	72
3.	Claim 2, 9, 16: “wherein each of the first plurality of cores and each of the second plurality of cores further include a respective caching layer therein.”	92
4.	Claims 3, 10, 17: “wherein each of the first plurality of cores includes therein a caching layer for data and a caching layer for instructions.”	94
5.	Claims 4, 11, 18: “wherein the core mix includes a state where some of, but not all of, the first plurality of cores and the second plurality of cores are enabled.”	96
6.	Claims 6, 13, 20: “wherein the first plurality of cores is at least three.”	97
7.	Claim 8	99
8.	Claim 15	105
F.	Ground 2B: Mathieson (EX-1003), Sakarda (EX-1007), and Liu (EX-1006) renders obvious claims 7, 14, and 21	107
1.	Claims 7, 14, 21: “coherency logic circuitry”	107
G.	Ground 2C: Mathieson (EX-1003), Sakarda (EX-1007), and Knebel (EX-1005) renders obvious claims 5, 12, and 19	108
1.	Claims 5, 12, 19: “a system memory interface to couple to a system memory to search for a cache line, requested by any core of the first plurality of cores and the second plurality of cores, that is not found in the caching layer” / “coupling a system memory interface of the multi-core	

processor to a system memory to search for a cache line,
requested by any core of the first plurality of cores and the
second plurality of cores, that is not found in the caching
layer” 108

IX. Secondary Considerations 114

X. Conclusion 115

I. INTRODUCTION

1. I, Murali Annavaram, submit this declaration to state my opinions on the matters described below.

2. I have been retained by MediaTek Inc. (“Petitioner”) as an independent expert in this proceeding before the United States Patent and Trademark Office. I am being compensated at my usual and customary rate of \$650 USD per hour. No part of my compensation depends on the outcome of this proceeding, and I have no other interest in this proceeding.

3. I understand that this proceeding involves U.S. Patent No. 10,740,281 (“the ’281 patent,” EX-1001), and I have been asked to provide my opinions as to the patentability of the claims of the ’281 patent.

4. I have been asked to consider the validity of certain claims of the ’281 patent based on certain prior art references. I have also been asked to consider the state of the art and prior art available as of December 22, 2011, which is the earliest claimed priority date on the face of the ’281 patent. Based on the combination of prior art references discussed in this declaration, it is my opinion that claims 1-21 of the ’281 patent are unpatentable for the reasons provided below.

II. QUALIFICATIONS AND BACKGROUND

5. I believe that I am well qualified to serve as a technical expert in this matter based upon my educational and work experience, which I summarize below.

6. I am an expert in the field of computer architecture and microarchitecture. My research focuses on efficient power management of chip multiprocessors (CMPs) and graphics processing units (GPUs), power efficient heterogeneous CMP designs, system-level power and performance tradeoffs, and reliability of computing platforms, superconducting CPU designs and efficient machine learning systems.

7. I am currently the Lloyd Hunt Chair professor in the Ming-Hsieh Department of Electrical and Computer Engineering, with a joint appointment in the Thomas Lord Computer Science department at the University of Southern California. I held the Dean's Chair Professorship from 2018-2021. I held the position of Robert G. and Mary G. Lane Early Career Chair at USC from 2012 to 2017. I started as a faculty member at USC in 2007.

8. I am founding and inaugural director of the USC-Meta center for Research and Education in AI and Learning at USC from 2022. I am the architecture thrust leader for the NSF's Expeditions in Computing center at USC focused on energy efficiency CPUs using superconducting technologies.

9. I co-authored *Parallel Computer Organization and Design*, a widely used textbook for graduate-level computer architecture courses. I teach several undergraduate and graduate computer architecture courses at USC, including topics such GPUs, out of order processor design, power and energy efficient design of

computer hardware, mobile and edge computing systems, and memory system design.

10. I received a Bachelor of Technology degree in Computer Science from the National Institute of Technology in Warangal, India, in 1993, and my Master of Science degree in Computer Science and Engineering from Colorado State University in 1996. I received my Ph.D. degree in Computer Science and Engineering from The University of Michigan in 2001. I worked as a Senior Research Scientist at Intel Corporation from 2001-2007 and a Visiting Faculty Researcher at Nokia Research Center in 2007 before joining the faculty of the University of Southern California in August 2007. I was a visiting faculty scientist at Meta during 2020-2021. During 2022-2023 I held the Rukmini Gopalakrishnchar visiting chair professorship at the Indian Institute of Sciences, Bengaluru, India.

11. I have received numerous awards including, for example, the IBM Faculty Partnership award in 2008, the Best Paper Award at the 2009 IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS) and the National Science Foundation CAREER award in 2010, Best Paper Award at the Federated Machine Learning workshop, Best paper and Best Student paper finalist nominations, Supercomputing 2019 conference. I am a Fellow of the Institute of Electrical and Electronics Engineers (IEEE) and a distinguished member of the Association of Computing Machinery (ACM).

12. I have had dozens of papers accepted on the topic of power efficient hardware design at major conferences in the field of computer architecture, including the International Symposium on Computer Architecture (ISCA), the High Performance Computer Architecture (HPCA), and the Symposium on Microarchitecture (MICRO).

13. My research on server energy efficiency has been selected for the IEEE Micro TopPicks award for the most influential computer architecture paper of the year 2013.

14. I have given keynote presentations at the Frontiers in Technology distinguished speaker series at the University of the California, Merced, 2021 Annual KIXX forum distinguished speaker series at the King Fahd university, Swedish Multicore Symposium in 2018 and at the IEEE International On-Line Test Symposium in 2014.

15. For my research and publications in this field, I was inducted into the ACM Special Interest Group on Microarchitecture Hall of Fame in December 2015. I am only one of 47 researchers over the past 48 years to receive this honor. I was also inducted into the IEEE Computer Society Technical Committee on Computer Architecture's High Performance Computer Architecture Hall of Fame. I am only one of 42 researchers over the past 22 years to receive this honor. Finally, I was inducted into the ACM International Symposium on Computer Architecture (ISCA)

Hall of Fame in June 2017. I am only one of 81 researchers to have received this honor over the past 44 years at the time of award selection.

16. I served as the General Co-Chair for ISCA 2018 and the Technical Program Chair for HPCA 2021, International Conference on Supercomputing 2024, Architecture Track Program Chair for HiPC 2014. I have also served as a program committee member for several international conferences and symposiums, including ISCA, MICRO, HPCA, SIGMETRICS, DSN, HiPC, and ASPLOS.

17. I served as an associate editor for the technical journal of the Association for Computer Machinery Transactions on Design Automation of Electronic Systems (ACM TODAES), and I served as a journal reviewer for the ACM Transactions on Computer Systems (ACM TOCS), ACM Transactions on Embedded Computing Systems (ACM TECS), ACM Computing Surveys, ACM Transactions on Architecture and Code Optimization (ACM TACO), and IEEE Top Picks in Microarchitecture 2010.

18. I am an inventor or co-inventor of seven issued patents, several of which relate to issues I discuss below.

19. I graduated 16 PhD students at USC, and 13 of them were solely advised by me. Of these, 6 are now in academia (both in the USA and abroad) as professors working on various computer architecture related problems.

20. My research group has published about 30 papers on the topic of CPU and GPU power optimized designs at top-tier computer architecture conferences in the past 10 years, where the typical acceptance rates are less than 20%. In particular, my group published 3 papers at the flagship computer architecture conference ISCA during 2016, which is a record for the highest number of published papers by a single research group at that time. These papers focused on improving the power and performance tradeoffs for various power efficiency techniques applied to highly parallel and heterogeneous processors such as CMPs, GPUs.

21. My research group has designed and evaluated the energy efficient techniques for heterogeneous CMPs, including building fully functional prototypes that required substantial understanding of the interactions between microarchitecture, circuits, operating systems and compilers. In particular, my group's work on KnightShift energy proportional computing developed an extensive set of power management schemes for heterogeneous computing platforms in active low-power state.

22. In my group we routinely work with various benchmarks, including SPEC, ParSec, to measure the power and performance impacts of architecture enhancements to chip designs. We also use various profiling tools, such as PIN and Vtune, for measuring various chip and system level activities. We use detailed

microarchitecture simulation tools, such as SimpleScalar, GPGPUsim, for power and performance evaluations.

23. My CV contains further details about my education, experience, publications, patents, and other qualifications to render an expert opinion.

24. I believe that I am well qualified to serve as a technical expert in this matter based upon my educational and work experience, which I summarize above.

25. Additional details about my employment history, fields of expertise, and publications are included in my curriculum vitae, which I understand is being submitted as EX-1023.

III. MATERIALS CONSIDERED

26. In forming my opinions, I have reviewed the following documents:

Exhibit	Description
EX-1001	U.S. Patent No. 10,740,281 (“the ’281 patent”)
EX-1002	U.S. Patent Publication No. 2008/0288748 to Sutardja
EX-1003	U.S. Patent Publication No. 2011/0213950 to Mathieson
EX-1004	David A. Patterson et al., <i>Computer Architecture: A Quantitative Approach</i> (2nd ed. 1996) (“Patterson-1”)
EX-1005	U.S. Patent Publication No. 2009/0106494 to Knebel
EX-1006	U.S. Patent Publication No. 2006/0053258 to Liu
EX-1007	U.S. Patent No. 9,442,758 to Sakarda
EX-1008	File History of U.S. Patent No. 10,740,281 (excerpted)
EX-1009	Lucien M. Censier et al., <i>A New Solution to Coherence Problems in Multicache Systems</i> , <i>IEEE Transactions on Computers</i> , Vol. C-27, No. 12 (Dec. 1978)
EX-1010	Homayoon Akhiani, et al., <i>Cache Coherence Verification with TLA+</i> , <i>Compaq Computer Corporation</i> , in <i>FM’99 – Formal Methods</i> (J.M. Wing et al eds., 1999)

Exhibit	Description
EX-1011	David A. Patterson et al., Computer Organization and Design – The Hardware/Software Interface (2nd ed. 1998) (“Patterson-2”)
EX-1012	Enric Herrero, <i>Distributed Cooperative Caching</i> , 2008 International Conference on Parallel Architectures and Compilation Techniques (PACT), Oct. 2008.
EX-1013	Allen B. Tucker, Computer Science Handbook (2nd ed. 2004)
EX-1014	Balaram Sinharoy et al., <i>IBM POWER7 Multicore Server Processor</i> , 50 IBM J. Res. & Dev. 3, Paper 1 (June 2011)
EX-1015	Alexandra Fedorova, et al., <i>Maximizing Power Efficiency with Asymmetric Multicore Systems</i> , 7 ACM Queue 10 (Nov. 20, 2009)
EX-1016	Mahmoud Naghibzadeh, Operating Systems Concepts and Techniques (2005)
EX-1017	Jim Handy, The Cache Memory Book (2nd ed. 1998)
EX-1018	U.S. Patent No. 6,801,984 to Arimilli
EX-1019	U.S. Patent Publication No. 2009/0089510 to Lee
EX-1020	U.S. Patent Publication No. 2009/0309243 to Carmack
EX-1021	Nvidia, <i>Variable SMP – A Multi-Core CPU Architecture for Low Power and High Performance</i> (2011)

IV. LEGAL STANDARDS

27. In considering the subject matter of the '281 patent and its claims in light of the prior art and forming my opinions, I am relying on certain legal principles that counsel in this case explained to me. My understanding of these concepts is summarized below.

28. I understand that the claims define the invention. I also understand that an unpatentability analysis is a two-step process. First, the claims of the patent are construed to determine their meaning and scope. Second, after the claims are construed, the content of the prior art is compared to the construed claims.

29. I understand that an invention, as defined by the claims of the patent, must not be anticipated or rendered obvious by the prior art.

A. Claim Construction

30. I understand that the United States Patent and Trademark Office interprets claim terms in an *inter partes* review proceeding under the same claim construction standard that is used in a United States federal court. I understand that under this standard, the meaning of claim terms is considered from the viewpoint of a person of ordinary skill in the art (“POSITA”) at the time of the alleged invention.

31. I have been informed that claim terms are generally given their ordinary and customary meaning as understood by one of ordinary skill in the art in light of the specification and the prosecution history pertaining to the patent. I understand, however, that claim terms are generally not limited by the embodiments described in the specification.

32. I understand that in addition to the claims, specification, and prosecution history, other evidence may be considered to ascertain the meaning of claim terms, including textbooks, encyclopedias, articles, and dictionaries. I have been informed that this other evidence is often less significant and less reliable than the claims, specification, and prosecution history.

B. Obviousness Under 35 U.S.C. § 103

33. I understand that a patent claim is invalid as obvious if the claimed invention would have been obvious to a person of ordinary skill in the art at the time the claimed invention was made. This means that even if all of the limitations of the claim cannot be found in a single prior art reference that would anticipate the claim, a person of ordinary skill in the art who knows about all the prior art would have come up with the claimed invention. I understand that in an obviousness determination, the person of ordinary skill in the art is presumed to have knowledge of all material prior art. I understand that whether a claim is obvious is based upon the determination of several factual issues.

34. In considering obviousness, I understand that one must determine the scope and content of the prior art. I understand that, in order to be considered prior art to a patent, a reference must be reasonably related to the claimed invention of that patent. A reference is reasonably related if it is in the same field as the claimed invention or is from another field to which a person of ordinary skill in the art would look to solve a known problem.

35. I understand that one must determine what differences, if any, existed between the claimed invention and the prior art.

36. I understand that a patent claim composed of several limitations is not proved obvious merely by demonstrating that each of its limitations was

independently known in the prior art. In evaluating whether such a claim would have been obvious, one may consider whether a reason has been identified that would have prompted a person of ordinary skill in the art to combine the elements or concepts from the prior art in the same way as in the claimed invention.

37. I understand that one may consider whether (1) the change is merely the predictable result of using prior art elements according to their known functions or the result of true inventiveness; (2) there is some teaching or suggestion in the prior art to make the modification or combination of limitations claimed in the patent; (3) the innovation applies a known technique that had been used to improve a similar device or method in a similar way; (4) the claimed invention would have been obvious to try, meaning that the claimed innovation is one of a relatively small number of possible approaches to the problem that could have been used by those of ordinary skill in the art with a reasonable expectation of success; (5) the invention merely substitutes one known element for another known element in order to obtain predictable results; (6) the invention merely applies a known technique to a known device, method, or product to yield predictable results; or (7) known work in the field may have prompted variations of use of the same inventions in the same or different fields due to market forces or design incentives that would have been predictable to a person of ordinary skill in the art.

38. I understand that one must be careful not to determine obviousness using the benefit of hindsight; many true inventions might seem obvious after the fact. I understand that the determination should be based on the position of a person of ordinary skill in the art at the time the claimed invention was made and should not reflect what is known today or what is learned from the teaching of the patent.

C. Secondary Considerations of Non-Obviousness

39. I understand that certain secondary considerations may be considered in evaluating obviousness in order to prevent hindsight bias. These secondary considerations include the commercial success of products that practice the patent, long-felt need for the patented technology, failure by others to solve the problem addressed by the patent, initial skepticism by others in the industry, industry recognition and praise of the patented products, and efforts by others to copy the patented technology.

40. I have been informed that these secondary considerations of non-obviousness must be balanced against the strength of the *prima facie* case of obviousness. Where the invention represents no more than the predictable use of prior art elements according to their established functions, secondary considerations are inadequate to establish non-obviousness.

41. I have also been informed that the patentee must establish a nexus between any secondary considerations and the claimed invention. Where a

secondary consideration results from something other than what is both claimed and novel in the patent, there is no nexus to the merits of the claimed invention.

42. I have been informed that, for example, the nexus requirement as it applies to evidence of commercial success requires that the patentee must show that the sales of a product that allegedly practices the claimed invention must be a direct result of the unique characteristics of the claimed invention, as opposed to other economic and commercial factors that are unrelated to the patented invention. I understand this to mean that if the product's commercial success is due to an element that existed in the prior art or that is not claimed by the patent, then no nexus exists.

43. I have been further informed that, for example, the nexus requirement as it applies to the long-felt need element must likewise be supported by evidence. Where the differences between the prior art and the claimed invention are minimal, it cannot be said that any long-felt need was met by the patented invention or that the patent solved any technological problems that were unaddressed by others.

44. I have been informed that evidence of industry praise must also show a nexus with the claimed invention, and that self-serving statements by the patentee regarding the merits of the claimed invention are generally not given weight.

V. THE '281 PATENT

A. Overview of the '281 Patent

45. The '281 patent discloses an “asymmetric performance multicore architecture with same instruction set architecture (ISA).” EX-1001, 1:23-26. Figure 4 shows low power (LP) cores 401 designed to be “lower performance and therefore consume less power” than cores 402. *Id.*, 3:58-4:3. The LP cores 401 “support[] the same instruction set 403” as the higher power (HP) cores 402. *Id.*

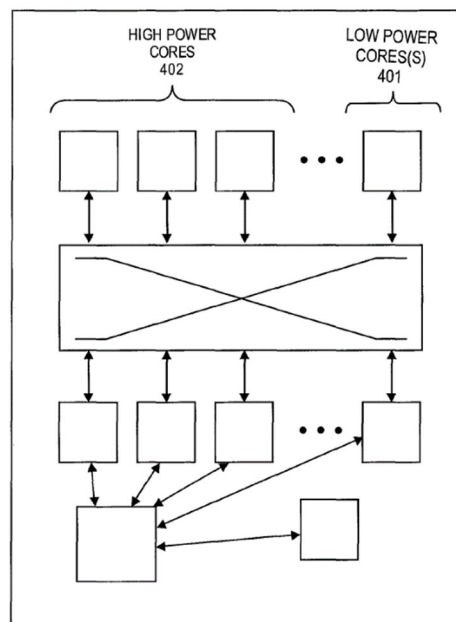


FIG. 4

Id., Fig. 4.

46. The '281 patent discloses several known features of multi-core processors. *See id.*, 1:30-2:3, Figs. 1, 2. This includes a multicore processor 100_1 with cores (101_1 through 101_N) connected to caching layers (103_1 through

103_N) and memory (105_1) for storing data/instructions. *Id.* According to the patent, power management schemes in the prior art include those operating on cores of either “identical” or alternatively cores of “radically different” designs in the same processor. *Id.*, 3:42-57.

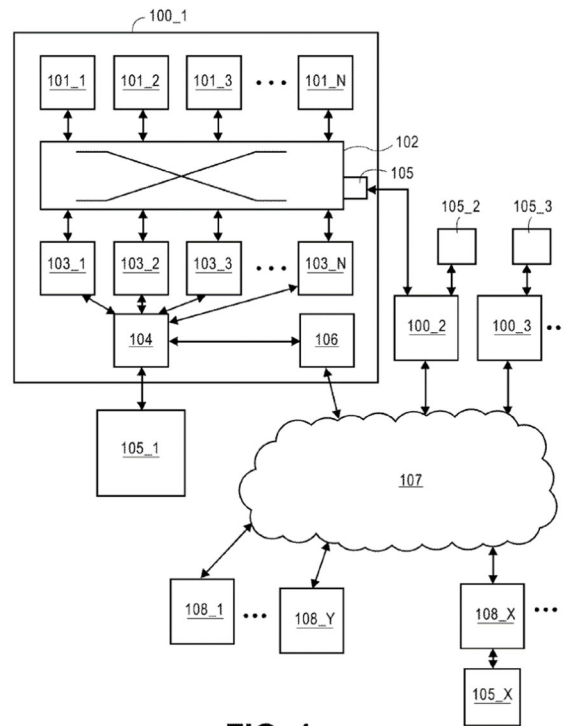


FIG. 1

Id., Fig. 1 (depicting prior art multi-core processors).

B. Summary of the Prosecution History of the '281 Patent

47. I have reviewed the prosecution history of the application that led to the '281 patent. EX-1008. During prosecution, the claims of the '281 patent were allowed after the Applicant amended the independent claims to recite “each of the second plurality of cores hav[ing] a maximum operating frequency that is less than

a maximum operating frequency of each of the first plurality of cores” in response to an obviousness rejection. EX-1008, 43-51, 30-39, 9-25.

C. Claim Construction of Terms in the '281 Patent

48. As discussed above, I have been informed that, for purposes of *inter partes* review, the standard for construction of terms in a patent’s claims is the same as that applied in federal district court litigation. I understand that a district court has yet to construe the claim terms of this patent. At this time, I do not see the need to construe any terms of the challenged claims for the purposes of this proceeding. Accordingly, in preparing my analysis I have applied the ordinary and customary meaning that each claim term would have to a person of ordinary skill in the art in light of the '281 patent’s specification and prosecution history. If any claim terms are differently construed, I reserve the right to reassess my opinions under said constructions.

VI. PERSON OF ORDINARY SKILL IN THE ART

49. I have been informed that patentability must be analyzed from the perspective of a “person of ordinary skill in the art” in the same field as the patent-in-suit at the time of the invention. I also understand that several factors are considered in assessing the level of ordinary skill in the art, including (1) the types of problems encountered in the art; (2) the prior art solutions to those problems;

(3) the rapidity with which innovations are made; (4) the sophistication of the technology; and (5) the education level of active workers in the field.

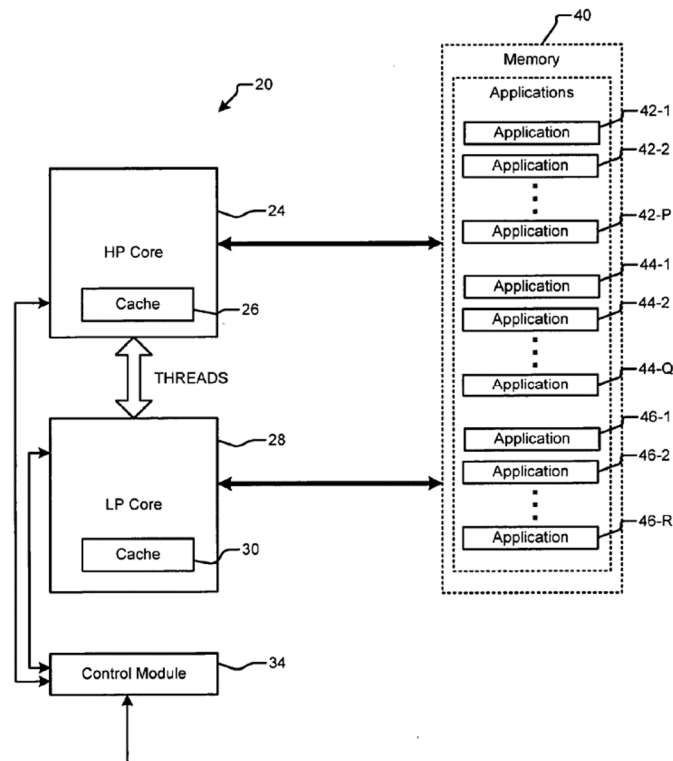
50. I refer to a person of ordinary skill in the art here as a “POSITA.” A POSITA at the time of the invention would have had a minimum of a bachelor’s degree in electrical engineering, computer engineering, physics, or a similar field, and approximately two years of industry or academic experience in a field related to microelectronic architecture design and development, including experience with power management techniques for multicore processors. Work experience can substitute for formal education and additional formal education can substitute for work experience. I at least meet this definition of a POSITA, now and as of the priority date of the ’281 patent.

51. I am also familiar with the knowledge of a POSITA as of the priority date of the ’281 patent, as I have taught and interacted with hundreds of such engineers both in industry and academia as of the priority date. I am able to opine on how a POSITA would have understood the disclosure and claims of the ’281 patent, the disclosures of the prior art, the various motivations to combine the prior art, and what combinations would have been obvious to a POSITA.

VII. THE PRIOR ART

A. Overview of Sutardja (EX-1002)

52. Sutardja discloses a multi-core processing system with a lower power (“LP”) core and a high power (“HP”) core. EX-1002 (“Sutardja”) ¶¶221-22, Figs. 3A, 3B.



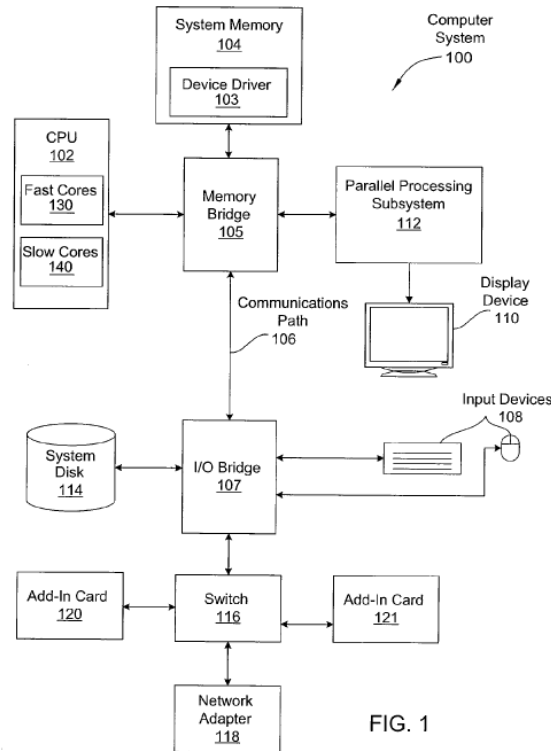
Id., Fig. 3A.

53. The LP cores are “low speed, low-power,” and the HP cores are “high-speed, high-power,” where the “transistors 106 of the HP core 24 tend to consume more power during operation in the active state than the transistors 110 of the LP core 28.” *Id.* ¶¶212, 230. HP and LP cores include caches 26 and 30, respectively. *Id.* ¶222, Fig. 3A. The system switches between cores “to reduce power consumption

while still supporting processing at high speed as needed.” *Id.* ¶ 222. Both HP and LP cores are “capable of running the same operating system and/or applications,” including “execut[ing] instructions, states, and/or threads of each other.” *Id.* And “LP core 28 may use an ISA [(instruction set architecture)] that is the same as or a subset of the ISA of the HP core 24.” *Id.* ¶225.

B. Overview of Mathieson (EX-1003)

54. Mathieson discloses a computer system 100 with fast cores 130 and slow cores 140, where “cores 130 are associated with higher performance and higher leakage power than the cores 140.” EX-1003 (“Mathieson”) ¶20, Fig. 1. The fast and slow cores “may implement substantially the same functionality,” or slow cores “may be capable of a subset of the functionality of the [fast] cores.” *Id.* ¶26. Mathieson’s designs, like Sutardja’s, decrease total power consumption, improve parallelism, and improve system efficiency. *Id.* ¶11, 34-35, 38.



Id., Fig. 1.

C. Overview of Patterson-1 (EX-1004)

55. Patterson-1 discloses techniques for storing data in caches. Its cache can be divided into an instruction cache and a data cache. EX-1004 (“Patterson-1”), 382. Patterson-1 explains that a data cache “cannot supply all the memory needs of the processor: it also needs instructions,” and “[a]lthough a single cache could try to supply both, it can be a bottleneck.” *Id.* Separate caches “also offer the opportunity of optimizing each cache separately: different capacities, block sizes, and associativities may lead to better performance.” *Id.*

D. Overview of Knebel (EX-1005)

56. Knebel discloses a multi-core architecture with first-level L1 caches and a shared, second-level L2 cache connected to storage 120. EX-1005 (“Knebel”) ¶10, Fig. 1. A cache controller 101 searches the L1 and/or L2 caches to “locate a line having a target address.” *Id.* ¶20. If the L1 cache “does not contain the address (a cache ‘miss’), the core then determines whether the next-fastest cache which it may access, the L2 cache 118.” *Id.* ¶12.

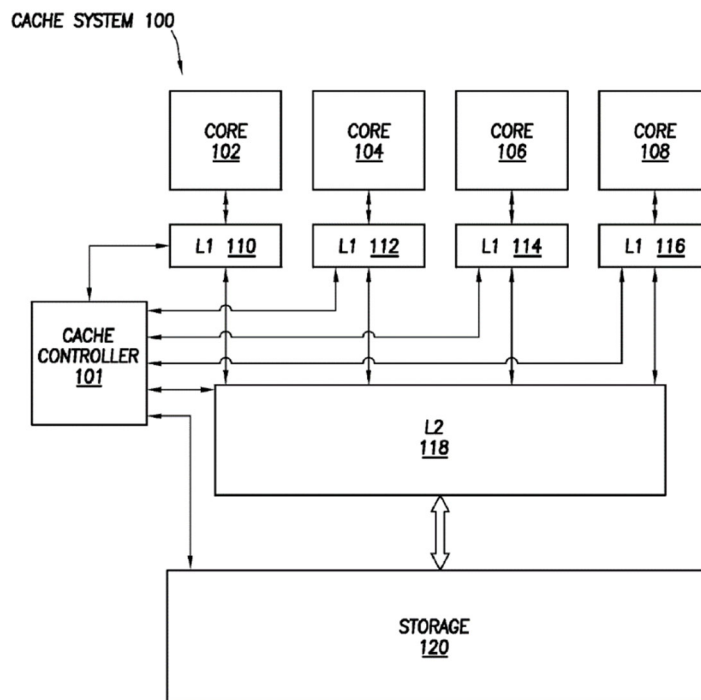


FIG. 1

Id., Fig. 1.

E. Overview of Liu (EX-1006)

57. Liu discloses methods of managing cache snoops “within a processor having a number of processor core caches and an inclusive shared cache.” EX-1006

(“Liu”) ¶¶3, 16. Liu’s techniques reduce traffic on processor core internal buses by “reducing the number of snoops from both external sources and other cores within a multi-core processor.” *Id.* ¶17.

F. Overview of Sakarda (EX-1007)

58. Sakarda discloses a multi-core processor system 100 with a first core executing at a first processing rate, and a second core executing at a different rate to conserve power. EX-1007 (“Sakarda”), Abstract, 1:26-35, 1:44-57, 2:45-63, Fig. 1. The same “computer instructions 110” are executed “from one processor core to another.” *Id.*, 4:17-21, 8:57-64.

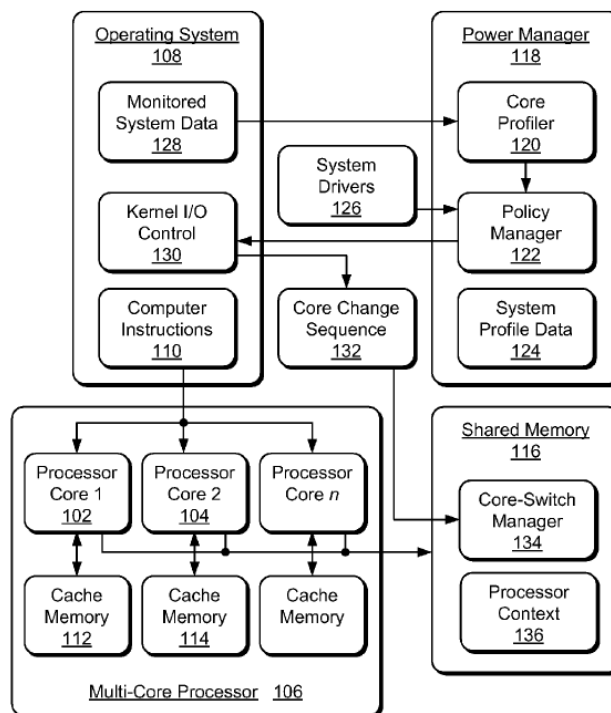


Fig. 1

Id., Fig. 1.

VIII. OPINIONS ON UNPATENTABILITY

In the analysis that follows, I identify the following prior art and combinations of prior art that, in my opinion, render obvious the claims of the '281 patent:

Ground	Prior Art	Basis	Claims
1A	Sutardja and Mathieson	§ 103	1, 2, 4, 6, 8, 9, 11, 13, 15, 16, 18, 20
1B	Sutardja, Mathieson, and Patterson-1	§ 103	3, 10, 17
1C	Sutardja, Mathieson, and Knebel	§ 103	5, 12, 19
1D	Sutardja, Mathieson, and Liu	§ 103	7, 14, 21
2A	Mathieson and Sakarda	§ 103	1-4, 6, 8-11, 13, 15-18, 20
2B	Mathieson, Sakarda, and Liu	§ 103	7, 14, 21
2C	Mathieson, Sakarda, and Knebel	§ 103	5, 12, 19

A. Ground 1A: Sutardja (EX-1002) and Mathieson (EX-1003) renders obvious claims 1, 2, 4, 6, 8, 9, 11, 13, 15, 16, 18, and 20

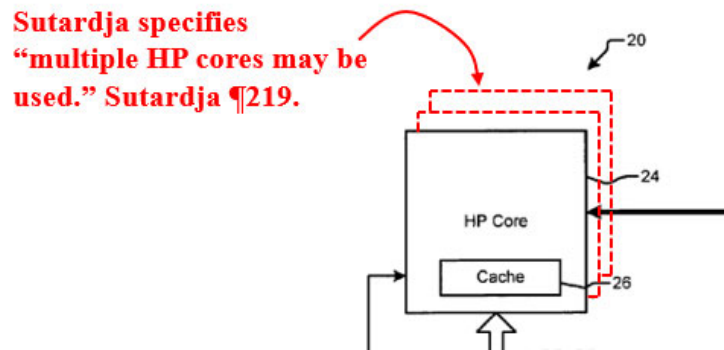
1. Motivation to Combine Sutardja and Mathieson

59. A POSITA would have found it obvious to combine Sutardja and Mathieson by using multiple, low-power cores, as Mathieson teaches, in Sutardja's architecture, which already employs multiple high-power cores and at least one low-power core. Mathieson teaches a multi-core processor architecture similar to Sutardja's that includes multiple, low-power cores 222. *See* Mathieson ¶25 ("The

CPU 102 includes a first set of cores 210, a second set of cores 220,” where “the first set of cores 210 includes one or more cores 212 ... and the second set of cores 220 includes one or more cores 222”), ¶28 (“[T]he cores 212 included in the first set of cores 210 comprise “fast” cores and the cores 222 included in the second set of cores 220 comprise “slow” cores”).

60. Mathieson explains that multiple “slow” cores enable processing of complex operations with less power, allow simpler switching of tasks from high to low power cores, and improve parallelism and performance. *Id.* ¶¶29, 34, 57. Following Mathieson’s teachings, a POSITA would have understood that, when applied to Sutardja, using multiple, low-power cores would have produced similar advantages, including using less power and improving parallelism. Indeed, a POSITA also would have understood that selecting the number of cores is a common design choice. *See* EX-1020 (“Carmack”) ¶28 (explaining “[t]he choice of the number of cores is a function of the tradeoff between the total area duplicated versus one or more other criteria” like the ones Mathieson identifies).

61. Sutardja expressly discloses that “multiple HP cores may be used.” Sutardja ¶219, Fig. 3A.



Id., Fig. 3A (annotated to depict Sutardja’s multiple HP cores and cropped).

62. The modified version of Sutardja’s Figures 3A and 5, shown below, illustrates the Sutardja-Mathieson combination with multiple HP and LP cores:

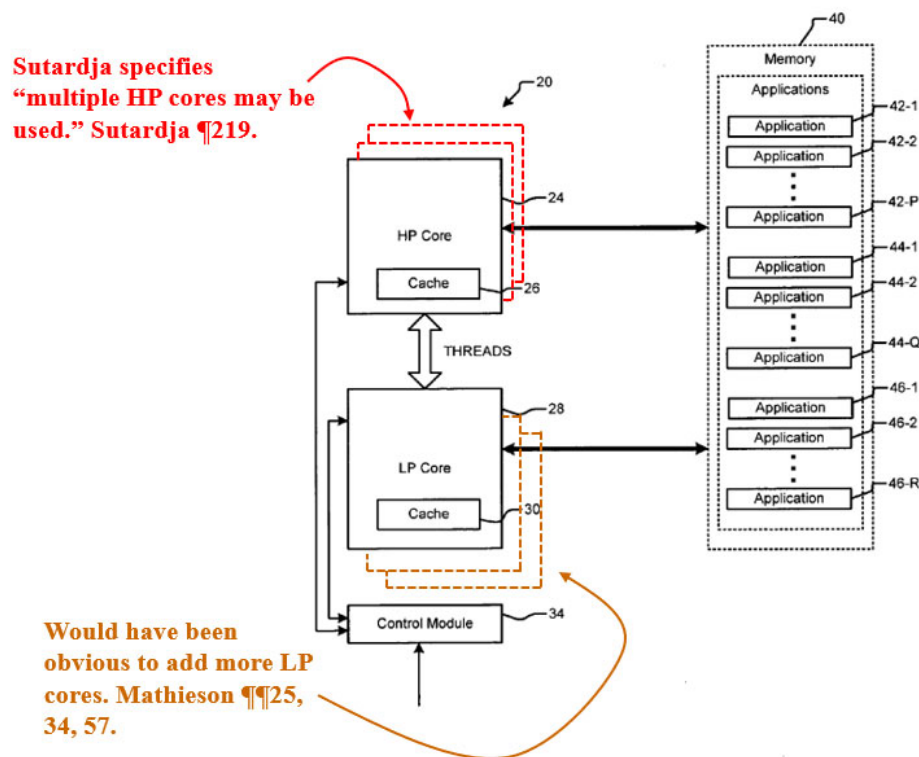


FIG. 3A

Sutardja, Fig. 3A (annotating multiple HP and LP cores).

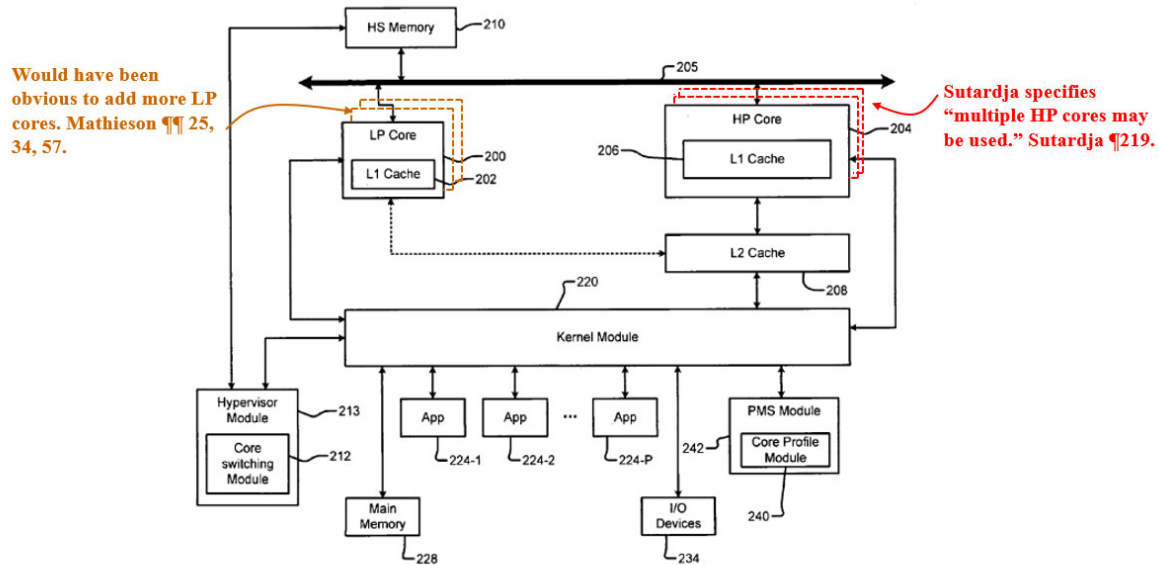


FIG. 5

Sutardja, Fig. 5 (annotating multiple HP and LP cores).

63. A POSITA would have been motivated to apply Mathieson's teachings to Sutardja for several reasons. First, Mathieson and Sutardja address similar problems, including power consumption in mobile devices. Sutardja ¶¶4-5 (addressing problems in "[m]obile computing devices such as cellular phones"); Mathieson ¶¶5-7 (explaining that, "[w]ith the proliferation of battery-powered mobile devices, efficient power management is quite important to the success of a product or system.")

64. Second, a POSITA would have recognized that the combination would have required nothing more than combining prior-art elements (e.g., Sutardja's basic architecture and Mathieson's teaching to use multiple, low-power cores) according to known methods (e.g., instantiating multiple, low-power cores instead of just one)

to yield predictable results (e.g., a chip with multiple, low-power cores to increase the processing capacity at lower power, increase parallelism, and enable simpler switching).

65. Third, a POSITA would have understood that using Mathieson's multiple low-power cores to improve Sutardja's basic architecture would have required nothing more than using a known technique to improve similar devices in the same way (e.g., by adding one or more low-power cores to Sutardja's architecture).

66. Fourth, prior-art teachings, such as those in Mathieson and Carmack (discussed above), would have led a POSITA to modify Sutardja to arrive at the claimed invention.

67. Fifth, a POSITA also would have recognized that, at times, there may be more demand than one slow core can manage, even from applications that do not require high-speed processing. Because slow cores consume less power than fast cores, it would have been desirable to use an additional slow core rather than a fast core in such circumstances to consume less power.

68. Thus, a POSITA would have been motivated to modify Sutardja's multicore architecture by incorporating Mathieson's teaching to use multiple, low-power/slow cores. Duplicating Sutardja's low-power core would not have resulted

in unexpected results, and a POSITA would have had a reasonable expectation of success in the combination.

2. Claim 1

a. [1p] “A multi-core processor comprising:”

69. Sutardja-Mathieson satisfies this feature. Sutardja discloses a processing system 20 with multiple cores, including at least one high-power core 24 and at least one low-power core 28. Sutardja ¶212 (“[B]oth increased power savings and increased performance may be achieved in a mobile device by using a multi-core processing system with a low-speed, low-power (LP) core and a high-speed, high-power (HP) core.”), ¶¶219, 221-22, 225, Figs. 3A, 3B, 5. Sutardja’s processing system 20 is therefore “a multi-core processing system” with multiple cores. *See id.*

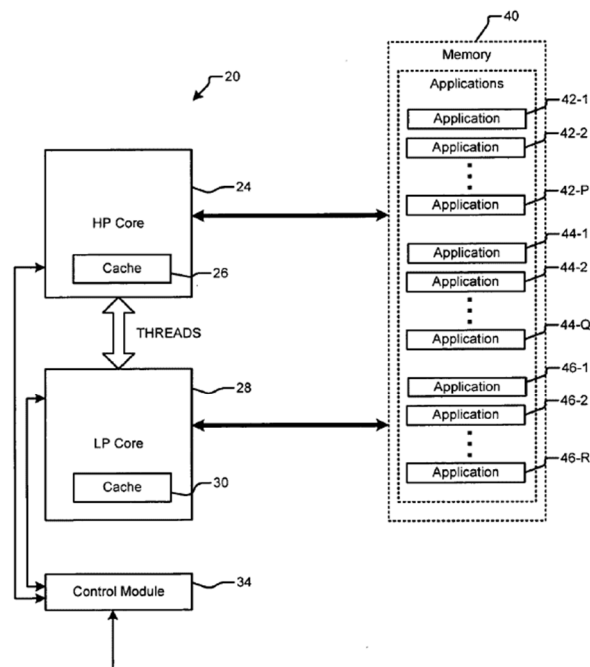


FIG. 3A

Id., Fig. 3A.

70. Similar to Sutardja, Mathieson discloses a multicore processor that has “**fast**” and “**slow**” cores. Mathieson, ¶25 (“The CPU 102 includes a first set of cores 210, a second set of cores 220, a shared resource 230, and a controller 240.”), ¶28 (“[T]he cores 212 included in the first set of cores 210 comprise ‘fast’ cores and the cores 222 included in the second set of cores 220 comprise ‘slow’ cores.”)

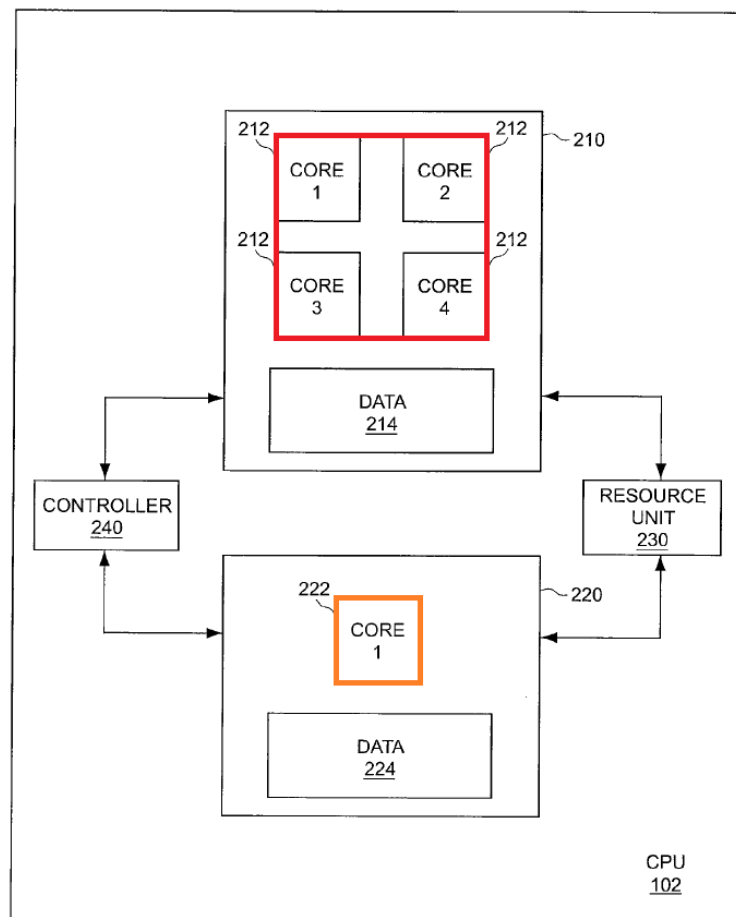


FIG. 2

Id., Fig. 2 (annotated).

71. Although Mathieson depicts a single core (“core 1”) in the second set of cores 222 in Figure 2, Mathieson discloses embodiments with *multiple* “slow” cores (e.g., **second set of cores 220**), *id.* ¶¶28-29, explaining multiple “slow” cores can process complex operations with less power, allow simpler switching of tasks from high to low power cores, and/or improve parallelism and performance data, *id.* ¶29 (“[T]he second set of cores 220 has a much lower leakage power loss than the first set of cores 210”), ¶ 34 (discussing “performance data” associated with the first set of cores and the second set of cores that includes “at least one of an operating frequency range of the first set of cores and an operating frequency range of the second set of cores, the number of cores in the first set of cores and the number of cores in the second set of cores, and an amount of parallelism between the cores in the first set of cores and an amount of parallelism between the cores in the second set of cores.”), ¶57 (discussing power advantages of low power cores: “operations should be processed by the second set of cores when less power would be consumed by the processing complex if the one or more operations were processed by the second set of cores.”).

72. A POSITA would have combined Mathieson’s use multiple low-power cores into Sutardja’s architecture for the reasons discussed above in the “Motivation to Combine Sutardja and Mathieson” section. Section VIII.A.1.

b. [1a] “a first plurality of cores and a second plurality of cores that support a same instruction set,”

73. Sutardja-Mathieson includes “a first plurality of cores and a second plurality of cores that support a same instruction set.” As depicted below, the Sutardja-Mathieson combination includes a first plurality of HP cores and a second plurality of LP cores. See [1p].

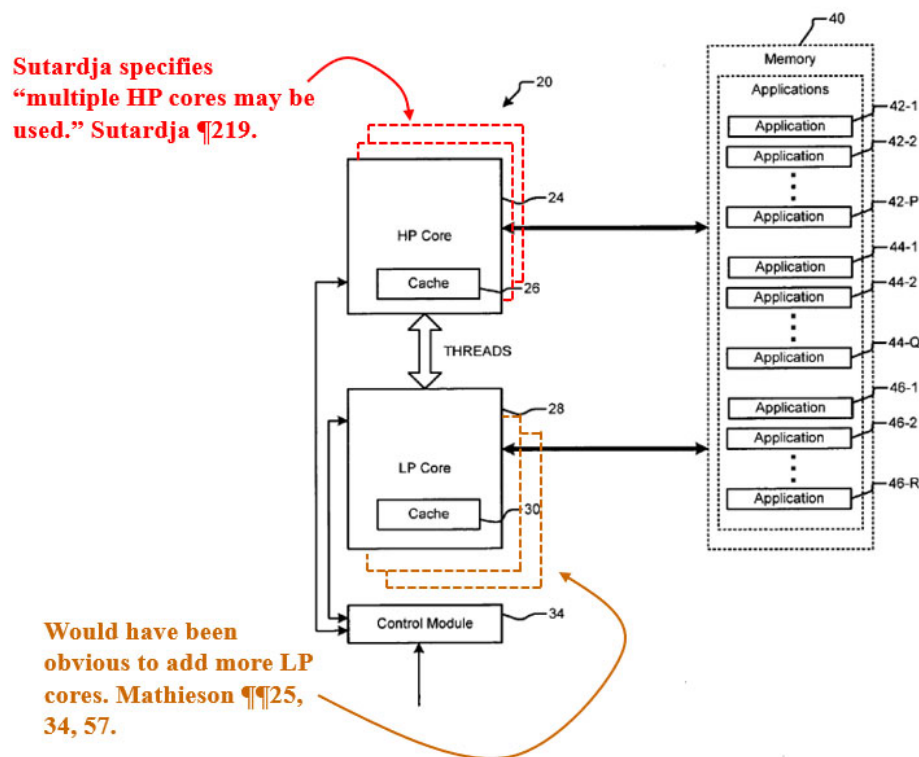


FIG. 3A

Sutardja, Fig. 3A (annotated).

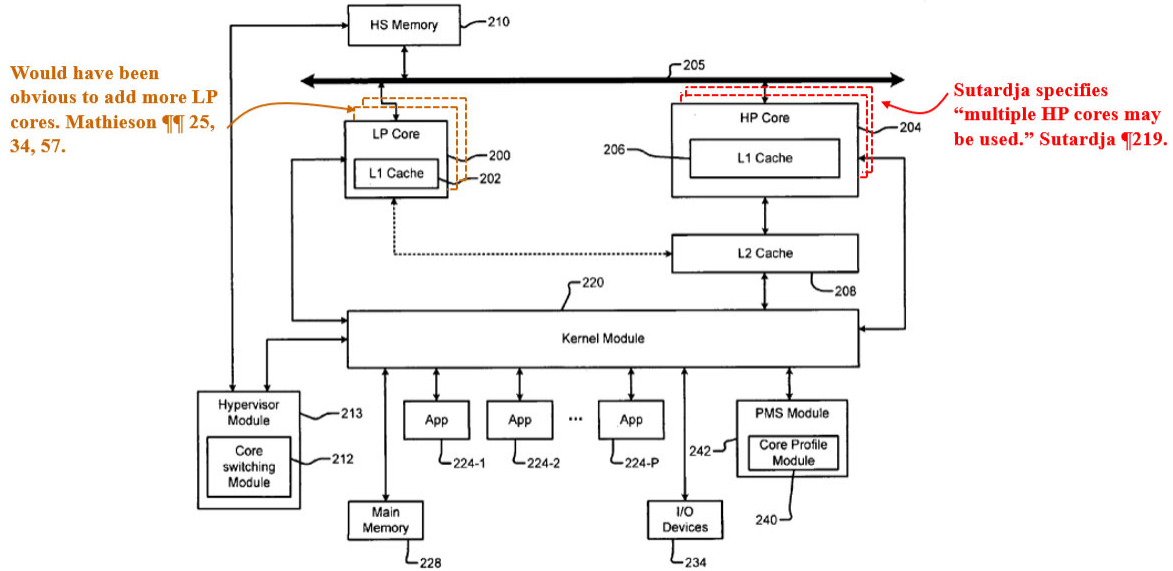


FIG. 5

Id., Fig. 5 (annotated).

74. Sutardja's high- and low-power cores use the **same instruction set architecture (ISA)**. Sutardja ¶225 ("The LP core 28 may use an ISA that is the same as or a subset of the ISA of the HP core 24."), ¶219 ("[W]hen execution is switched from one core to another, the other core executes instructions without translation," thereby indicating the same ISA); *see also id.* ¶¶222 ("Both HP and LP cores 24 and 28 are capable of running the same operating system and/or applications. Additionally, both HP and LP cores 24 and 28 can execute instructions, states, and/or threads of each other and continue processing."), ¶248.

75. Although Sutardja also states that "LP and HP cores 200 and 204 may execute ... different ISAs," *id.* ¶249, Sutardja's descriptions make it clear that both cores can run the same ISA, *id.* ¶219 ("[W]hen execution is switched from one core

to another, the other core executes instructions without translation”). Executing on two cores without needing a translation is possible primarily when they use the same ISA. Additionally, just because the cores *may* use different ISAs does not mean they *must* use different ISAs. Sutardja’s cores may *also* use the same ISAs. *See id.* ¶213 (“An instruction set architecture (ISA) of the LP core 12 may differ from the ISA of the HP core 16. . . . Alternately, both may use the **same ISA’s**.”) (emphasis added).

76. To the extent the ISAs may differ in Sutardja, Sutardja consistently teaches that the “first [ISA] set is a superset of the second [ISA] set.” *Id.* ¶¶14, 45, 75, 106. Thus, even in embodiments of Sutardja where the cores *may* use different ISAs, all cores still would support at least the same second set of instructions (because the first set of ISAs encompasses the second set of ISAs). All cores thus “support a same instruction set,” as claimed.

c. [1b] “wherein the first plurality of cores are higher performance and consume more power than the second plurality of cores, and”

77. Sutardja-Mathieson satisfies this feature. Sutardja discloses that **high-power cores 24** are high-speed and high-power while **low-power core 28** is low-speed and low-power, so the **high-power cores 24** are higher performance than the **low-power core 28**. Sutardja ¶¶211-213 (“[T]he LP and HP cores (hereinafter cores) 12 and 16 have different performance and capabilities (e.g., different processing speeds, power consumption, etc.).”), ¶230 (“The transistors 106 of the

HP core 24 tend to consume more power during operation in the active state than the transistors 110 of the LP core 28.”). Sutardja refers to speed as a proxy for performance, which is a very common practice for a POSITA. *See id.* ¶4 (disclosing that “capabilities of the processor such as processing speed are typically selected to match the performance requirements”).

78. Likewise, Mathieson teaches a set of **fast cores 210** with higher performance than a set of **slow cores 220**. Mathieson ¶¶28-29 (“The second set of cores 220 includes one or more ‘slow’ cores 222 constructed from slower transistors that are not capable of operating as quickly as the transistors includes in the cores 212 of the first set of cores 210. In some embodiments, the second set of cores 220 has a much lower leakage power loss than the first set of cores 210, but is not capable of achieving the same performance levels as the first set of cores 210.”).

79. Mathieson also teaches that, below a certain operating frequency, the system consumes less power using the **slow cores** than when using the **fast cores**. *Id.* ¶¶30-31, Fig. 6. Thus, when applying Mathieson’s teachings to Sutardja’s modified architecture, “**the first plurality of cores** are higher performance and consume more power than **the second plurality of cores**.”

- d. [1c] “each of the second plurality of cores have a maximum operating frequency that is less than a maximum operating frequency of each of the first plurality of cores;”

80. Sutardja-Mathieson satisfies “each of **the second plurality of cores** have **a maximum operating frequency** that is less than **a maximum operating frequency** of each of **the first plurality of cores.**” Sutardja teaches that its **high-power cores** (e.g., the **first plurality of cores**) can operate at **frequencies greater than 1 GHz**, while its **low-power core** (e.g., the **second plurality of cores**) operates at **frequencies less than 500 MHz**. Sutardja ¶234; *see also id.* ¶74 (“a first maximum speed ... greater than a second maximum speed”), ¶44 (selectively operating cores at different “maximum speed[s]”), ¶105 (same).

Sutardja specifies
HP cores have max
frequency over 1 GHz
Sutardja ¶230

Sutardja specifies LP
cores have max
frequencies of 500
MHz. Sutardja ¶212.

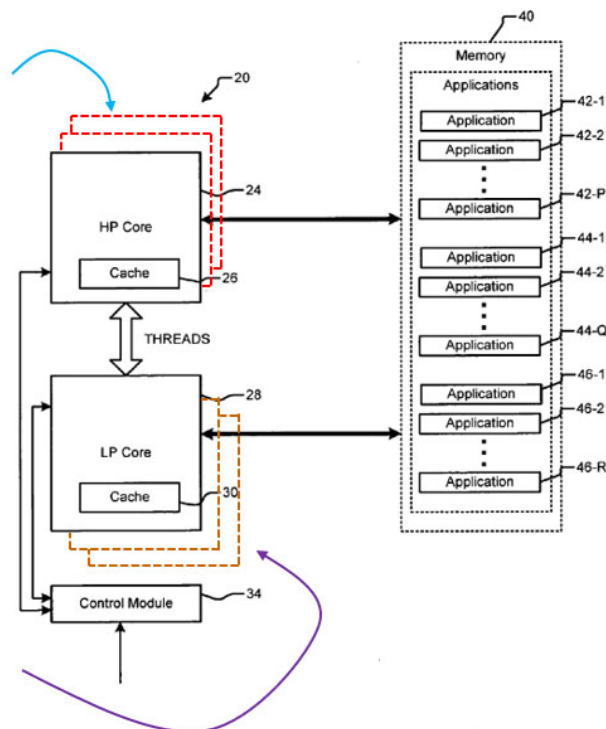


FIG. 3A

Id., Fig. 3A (annotated).

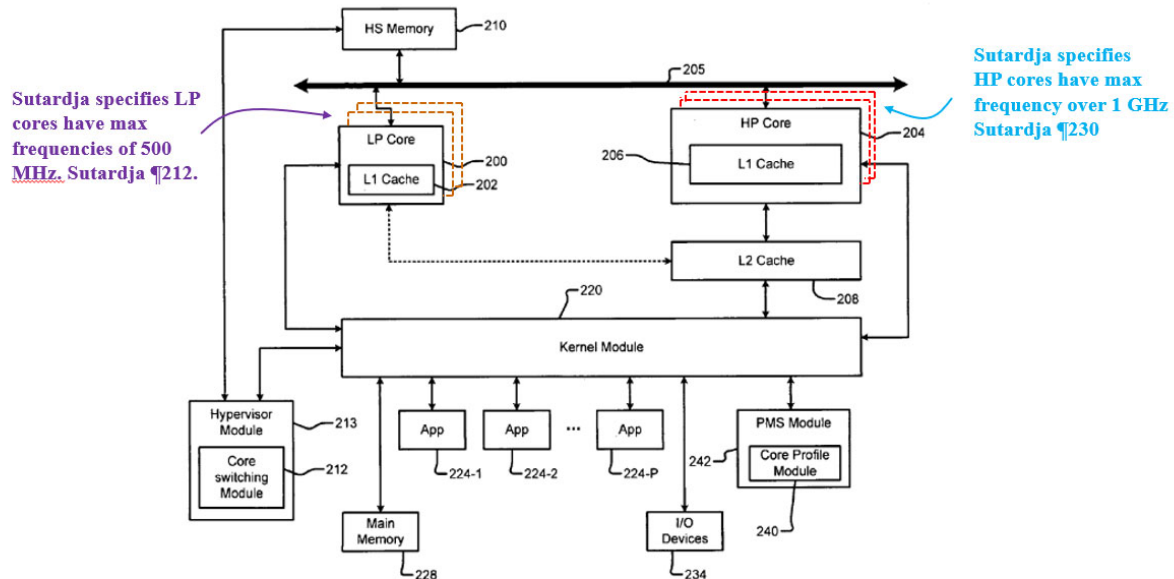


FIG. 5

Id., Fig. 5 (annotated).

- e. [1d] “a caching layer shared by the first plurality of cores and the second plurality of cores; and”

81. Sutardja-Mathieson satisfies “a **caching layer** shared by the **first plurality of cores** and **the second plurality of cores**.” Sutardja’s **high-power cores** and **low-power cores** may share an **L2 cache 206**, which serves as a last-level cache (LLC) for Sutardja’s various cores. Sutardja ¶248. The **L2 cache** is highlighted below.

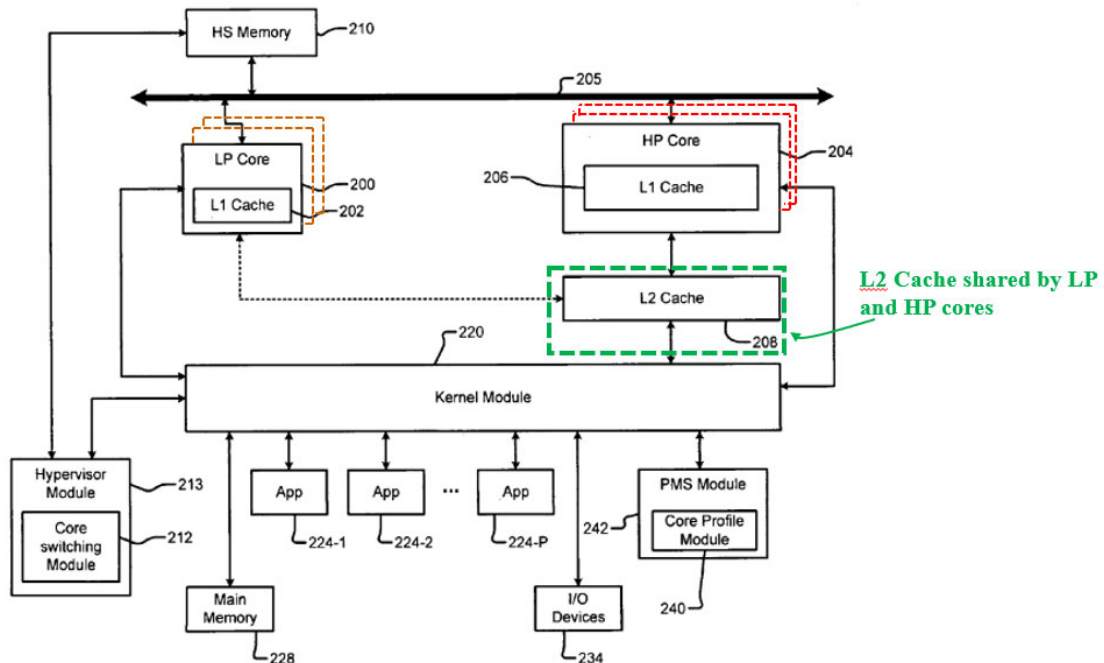


FIG. 5

Id., Fig. 5 (annotated).

82. A POSITA would have understood Sutardja's Figure 5 describes additional aspects of the same system as Sutardja's Figure 3A. For instance, Figure 3A includes a Control Module 34 "that switches cores of the multi-core processing system." Sutardja ¶221; *see also id.* ¶192, ¶228, Fig. 3A. Sutardja's Figure 5, in turn, describes additional details of an "exemplary control module with first and second cores." *Id.* ¶195. Although Sutardja introduces the basic functions of the multi-core control module in the context of Figures 3A-3D, *see id.* ¶228, ¶233, ¶¶235, it provides additional details about their structure and operation, including how they may be integrated with other system components, in the context of Figures 4-10. For instance, control module 34 of Figure 3A includes a core profile

module 120, a performance profile module 116, and an idle profile module 118. *See id.* ¶¶235-241, Fig. 3C. So does the control module of Figure 5. Specifically, it includes a PMS module 242, which is where the core profile module, performance profile module, idle profile module are located. *See id.* ¶251, ¶278, ¶280, ¶¶285-286, Figs. 5, 9A-9B.

83. A POSITA thus would have understood the multi-core control modules of Figures 3A and Figure 5 describe the same structure. In addition to these shared features, Sutardja explains the LP and HP cores 200 and 204 in Figure 5 may “have some or all of the attributes described herein,” including the LP and HP cores in Figure 3A, for instance. Sutardja ¶248. Given these similarities, as well as the overall organization of the Sutardja reference, a POSITA would have understood, or at least found it obvious, to take Sutardja’s descriptions of features in Figure 5 and to apply them in the context of Figure 3A, and vice versa. Indeed, Sutardja discloses that a POSITA would “appreciate from [Sutardja’s] description” that “the broad teachings of the disclosure can be implemented in a variety of forms,” and that “while [Sutardja’s] disclosure includes particular examples, the true scope of the disclosure should not be so limited since other modifications will become apparent to the skilled practitioner upon a study of the drawings, the specification, and the following claims.” *Id.* ¶328. This teaches and at least suggests applying the features of Sutardja’s Figure 5 and to Figure 3A, and vice versa.

84. Mathieson also teaches **fast cores 210** and **slow cores 220** sharing **resource unit 230**. Mathieson ¶25, Fig. 2. Like Sutardja, Mathieson discloses that **resource unit 230** can be a **shared L2 cache**. *Id.* ¶¶44, 45-46.

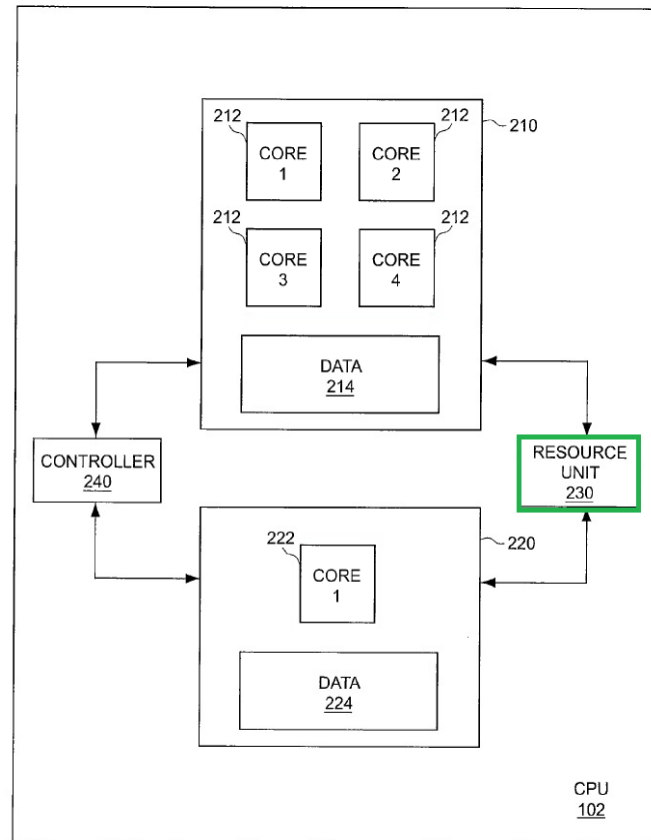


FIG. 2

Id., Fig. 2 (annotated).

- f. [1e] “power management hardware to enable and disable the first plurality of cores and the second plurality of cores,”

85. Sutardja-Mathieson includes “**power management hardware** to enable and disable **the first plurality of cores** and **the second plurality of cores**.”

86. Sutardja discloses a **Power Management System (PMS) module 242** that manages power consumption. Sutardja ¶251 (“A power management system (PMS) module 242 may be provided to manage power consumption and to initiate or control changes in system operation based on the monitored system loading.”). **PMS module 242** can trigger a core switching module 212 to enable/disable the **high-power** and **low-power** cores. *Id.* ¶¶218, 253, 271. The term “module” in Sutardja comprises “power management hardware,” as claimed, to perform the functions Sutardja describes. *Id.* ¶209; *see also* Mathieson ¶43 (“[C]ontroller 240 is configured to transfer the processor state from the first set of cores to the second set of cores . . . [and] triggers a hardware mechanism that stops and powers off the first set of cores 210, and boots the second set of cores 220.”).

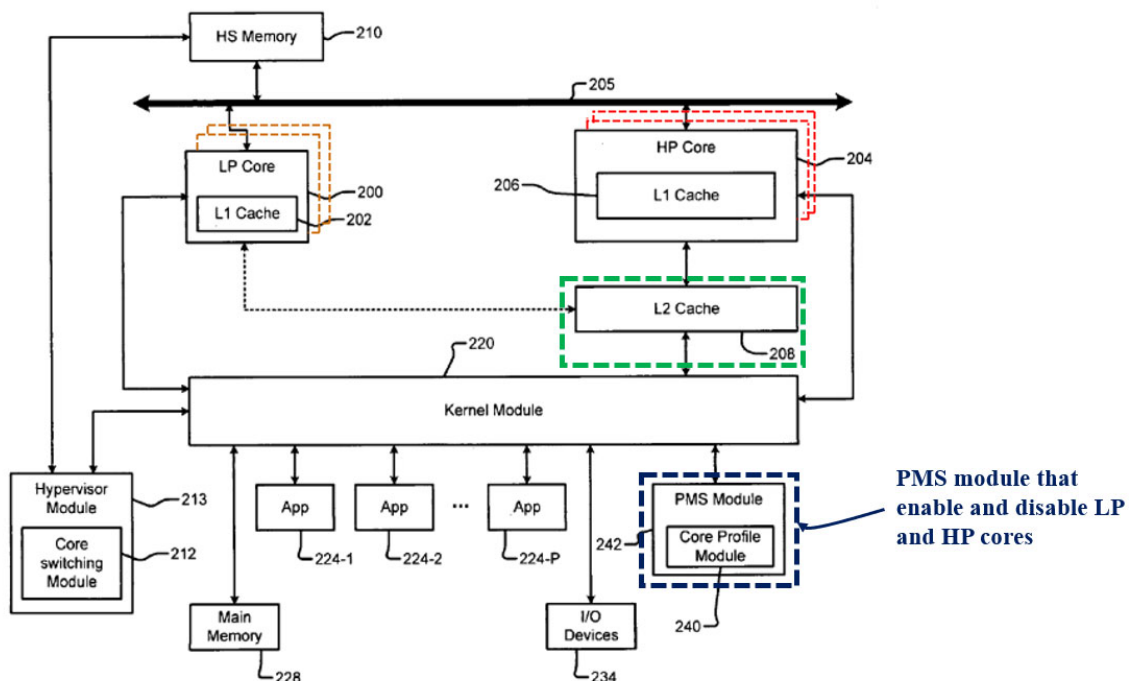


FIG. 5

Sutardja, Fig. 5 (annotated).

- g. [1f] “wherein an operating system that executes on the multi-core processor is to monitor a demand for the multi-core processor and control a core mix of the first plurality of cores and the second plurality of cores based on the demand with the power management hardware.”

87. Sutardja-Mathieson satisfies “wherein an operating system that executes on the multi-core processor is to monitor a demand for the multi-core processor and control a core mix of the first plurality of cores and the second plurality of cores based on the demand with the power management hardware.”

88. Sutardja discloses an operating system that executes on the multi-core processor by explaining that “LP and HP cores 200 and 204 may execute the same operating system (OS).” Sutardja ¶249. A “kernel module 220, which is a component of the OS, manages system resources.” *Id.* Kernel module 220 provides the lowest level abstraction of the resources and is coupled to resources like PMS module 242, which manages or controls power consumption, and core profile module 240, which is part of PMS module 242 and “may monitor system loading.” *Id.* ¶¶249, 251, Fig. 5. Sutardja therefore discloses an operating system that monitors a demand for the multi-core processor.

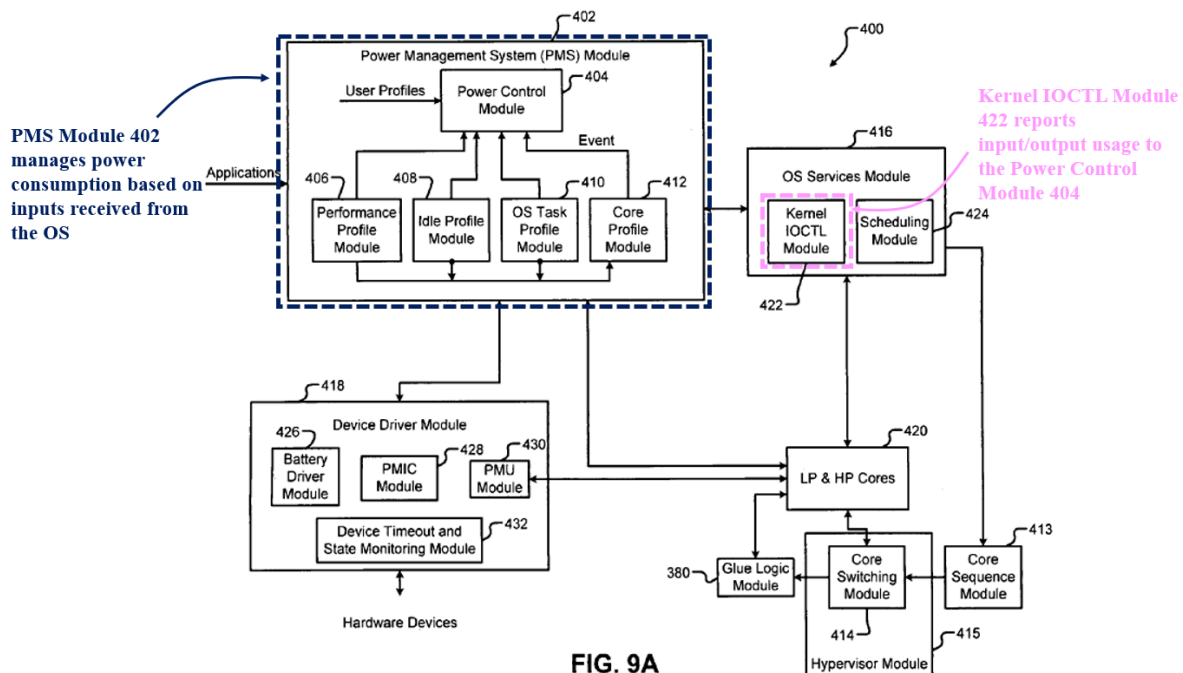
89. Sutardja also discloses an operating system that controls a core mix of processors. In Figure 9A, Sutardja discloses “an exemplary power management system that dynamically switches cores.” *Id.* ¶¶199, 278, Fig. 9A. A POSITA would

have understood this power management system would be applied to the multi-core processor described in Figures 3A and 5. Indeed, Sutardja disclose that the system of Figure 3A “may be useful in extending battery life in mobile devices that operate on battery power,” and in Figure 9A describes “a power management system (PMS) 400 that optimizes power consumption *of the mobile device* using dynamic core switching.” Sutardja ¶278 (emphasis added). Sutardja’s features are therefore parts of the same system in a mobile device to conserve battery life. Figure 9A shows **PMS module 402**, which a POSITA would have understood to be the same as (and describe additional details of) the **PMS module 242** in Figure 5. *See id.*, Fig. 9A, ¶278. The **PMS module 402**, like **PMS module 242**, manages power consumption based on various inputs, including “inputs received from the **OS**.” *Id.* ¶278; *see also id.* ¶¶251-253 (PMS module 242 managing power consumption based on “monitored system loading”).

90. Sutardja further teaches that performance profile module 406 (part of PMS module 242) “may generate outputs indicating current performance and demanded or desired performance for applications.” *Id.* ¶280. A POSITA would have understood Sutardja’s “demanded or desired performance for applications” refers to a system demand. Performance profile module 406 may “generate the outputs partly based on inputs received from an **OS services module 416**.” *Id.* **OS services module 416**, which “may comprise a **kernel input/output control**

(IOCTL) module 422 ... may report current I/O usage to the power control module 404.” *Id.* ¶281.

91. A POSITA would have understood **kernel IOCTL module 422** corresponds to **kernel module 220** of Figure 5. *See id.* ¶253. Because **kernel IOCTL module 422** “may report current I/O usage to the **power control module 404**,” *id.* ¶281, a POSITA would have understood **PMS module 404**, which includes performance profile module 406, determines a demand based on inputs from **kernel IOCTL module 422**. **PMS module 402** manages power consumption based on this demand. Sutardja ¶278. Annotated Figure 9A (below) highlights **kernel IOCTL module 422** and **PMS module 404**.



Id., Fig. 9A (annotated).

92. As discussed, a POSITA would have recognized that Figure 9A describes different aspects of the same processing system (e.g., in a mobile device) described in Figure 5. A POSITA thus would have understood **PMS module 242** (see Figure 5) receives inputs describing the demand in the system from **kernel module 220** (see Figure 9A) and manages power consumption based on that demand. *See also* Sutardja ¶¶252-255.

93. Additionally, a POSITA would have recognized that, via its connection to **PMS module 242**, **kernel module 220** controls the core mix by enabling and disabling **high power** and **low power** cores based on demand. *See* [1e]. For example, the multi-core processor may operate in a HP mode or LP mode, based on demand, where Sutardja discloses enabling and disabling the LP core and HP or maintaining “the LP core 28 . . . in the active state during all or part of the HP mode.” Sutardja ¶223. **Kernel module 220**, via the power management hardware, controls this core mix by changing operating frequencies based on demand. *Id.* ¶252. Annotated Figure 5 below highlights **kernel module 220** and its connection to **PMS module 242**.

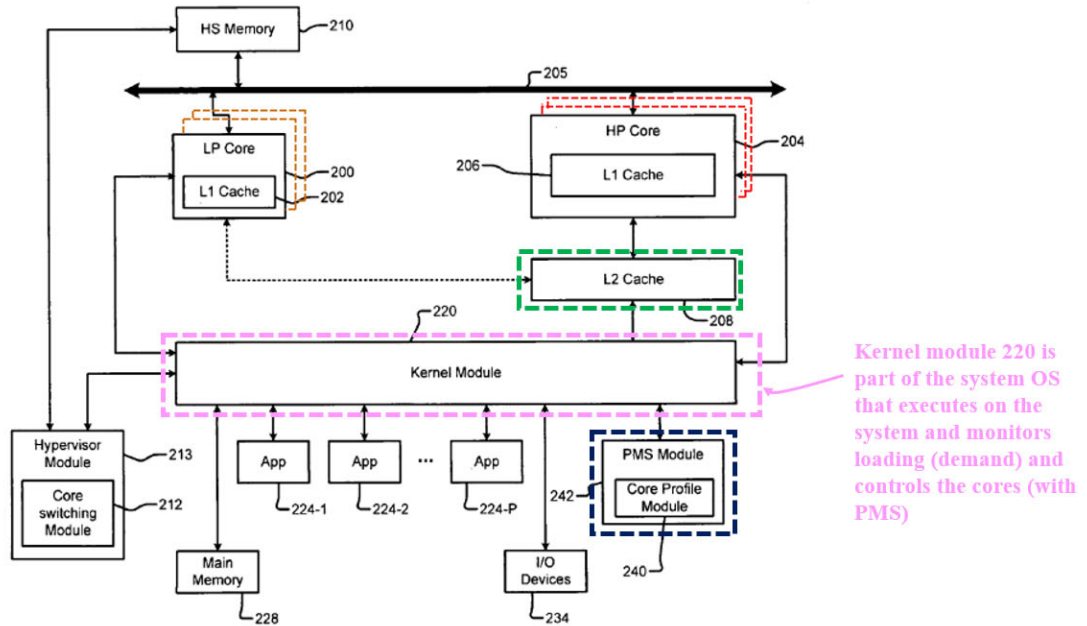


FIG. 5

Id., Fig. 5 (annotated).

3. **Claims 2, 9, 16: “wherein each of the first plurality of cores and each of the second plurality of cores further include a respective caching layer therein.”**

94. Sutardja-Mathieson satisfies this feature. Sutardja discloses that each core has its own L1 cache. Sutardja ¶¶222, 234, 247-48, Figs. 3A, 5. For example, Sutardja discloses an L1 cache 202 for LP core 200 and an L1 cache 206 for HP core 204. *Id.* ¶¶247-48, Fig. 5. Analogously, Sutardja discloses a cache 30 for LP core 28 and a cache 26 for HP core 24. *Id.* ¶¶222, 234, Fig. 3A.

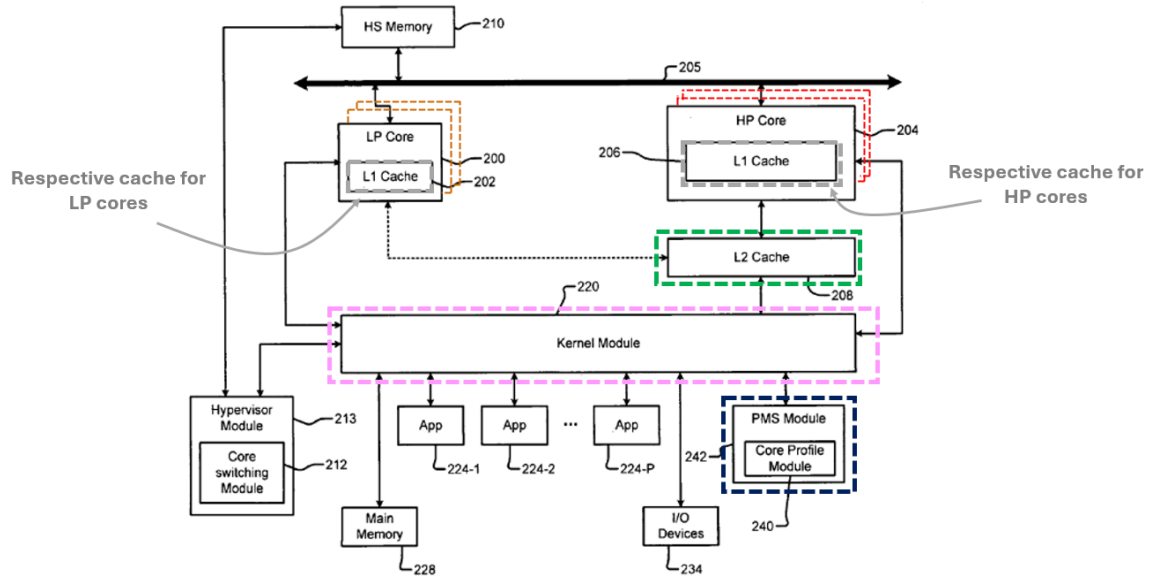


FIG. 5

Id., Fig. 5 (annotated).

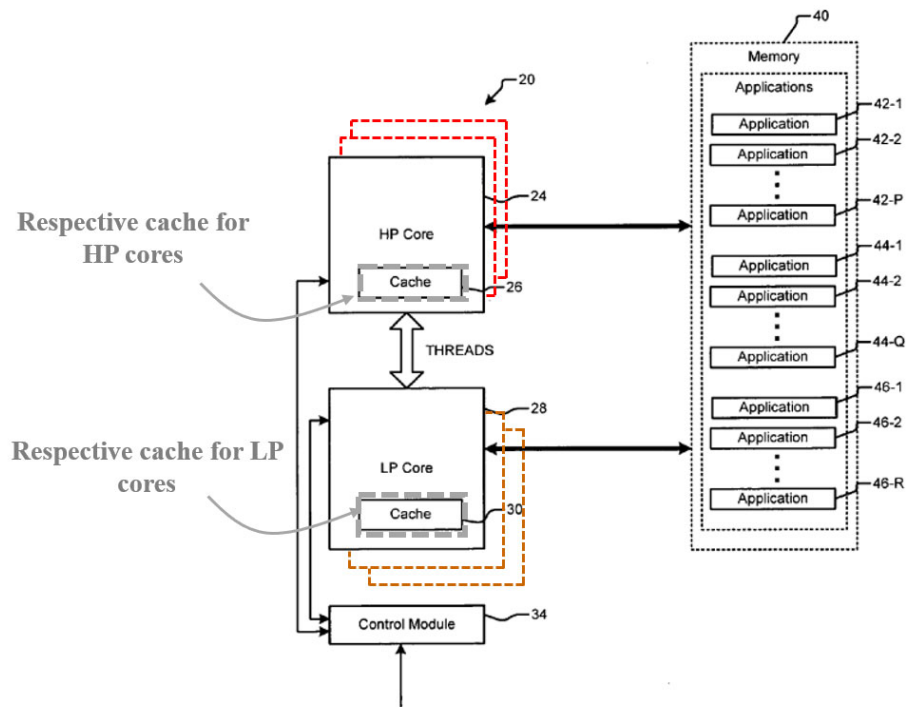


FIG. 3A

Id., Fig. 3A (annotated).

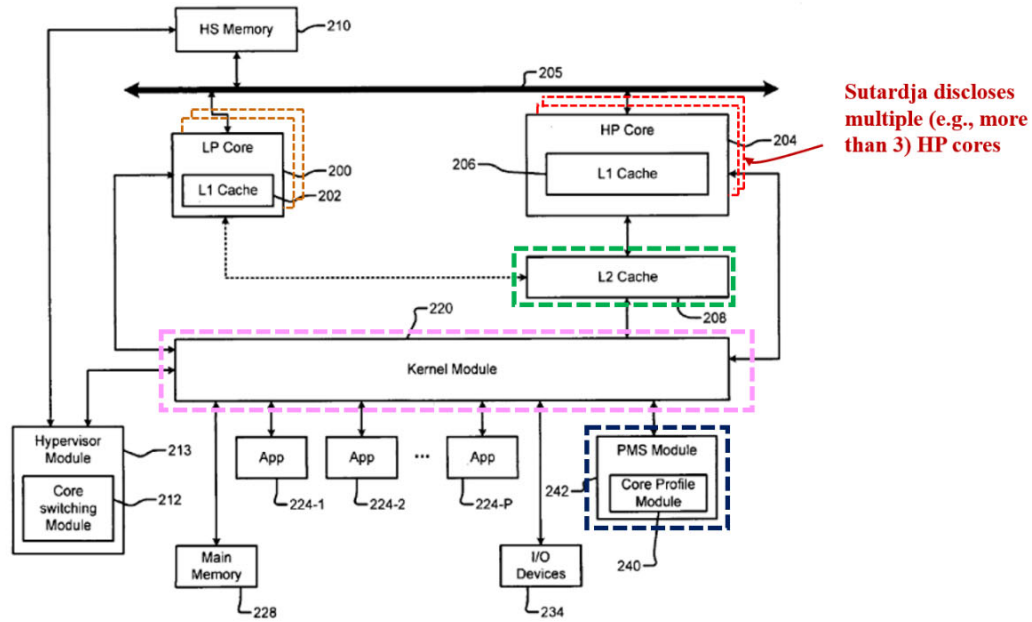
4. Claims 4, 11, 18: “wherein the core mix includes a state where some of, but not all of, the first plurality of cores and the second plurality of cores are enabled.”

95. Sutardja-Mathieson satisfies this feature. As discussed in [1f], Sutardja teaches the claimed “core mix” by disclosing that the **PMS module 242** enables/disables the **high power** and **low power** cores. *See supra* [1f].

96. Additionally, Sutardja discloses states in which some, but not all, of the cores are enabled. According to Sutardja, while one core is active another core may be disabled to reduce power consumption. Sutardja ¶218 (“In some implementations, only one core may be active (i.e., executing applications) at a time. When one core is active, other cores may be disabled (i.e., deactivated) to save power.”), ¶223 (“To reduce power, however, the LP core 28 may be operated in the inactive state while the HP core 24 is in the active state.”). This constitutes a state where some, but not all, of the first plurality of cores and the second plurality of cores are enabled, as claimed. Indeed, Sutardja corresponds with the ’281 patent. For example, the ’281 patent discloses embodiments in which its high power cores are enabled only after all low power cores are enabled. EX-1001, 4:55-6:31, Figs. 6, 7. The claims do not recite, nor does the specification disclose, enabling “some of, but not all of” *each* plurality of cores. Instead, Sutardja’s activating only its low power core satisfies this limitation because some but not all of the combination of cores are enabled.

5. Claims 6, 13, 20: “wherein the first plurality of cores is at least three.”

97. Sutardja-Mathieson satisfies this feature. Sutardja discloses having multiple **high-power cores**, where multiple identical cores may be used based on performance requirements. Sutardja ¶219 (“[W]hen execution is switched from one core to another, the other core executes instructions without translation. Accordingly, the switching of cores is transparent or invisible to the OS and applications, which continue to run as if the system comprises a single core instead of multiple cores”), ¶248 (“In some implementations, the HP core 204 may comprise multiple identical cores.”). Mathieson discloses at least three high-power cores because its “**fast**” plurality of cores includes four cores. Mathieson ¶28 (“In some embodiments, the first set of cores 210 may include multiple cores, e.g., four cores”), ¶57 (disclosing “four course included in the first set of cores”), Fig. 2 (four cores 212). Because Sutardja teaches multiple high-power cores and Mathieson teaches more than three “fast” high-power cores, a POSITA would have found it obvious to use at least three high-power cores in the Sutardja-Mathieson system.



Sutardja, Fig. 5 (annotated).

98. A POSITA would have found it obvious to use at least three high-power cores in Sutardja's architecture because such an implementation would have involved no more than combining prior-art elements (Sutardja's multiple high-power cores and Mathieson's four high-power cores) according to known methods (instantiating four cores in the device layout) to yield predictable results (improved processing capacity/performance). Sutardja's disclosure that multiple high-power processors may be used to improve performance and Mathieson's suggestion to use four high-power cores to improve processing capacity would have led a POSITA to modify Sutardja with Mathieson to implement Sutardja's architecture using four high-power cores. For instance, if demand exceeded what two high-power cores could accommodate, it would have been obvious to use at least three high-power

cores. *See* Sutardja ¶219. Doing so would have required nothing more than duplicating high-power cores with no unexpected results. *See* [1a]. A POSITA would have had a reasonable expectation of success in the combination.

6. Claim 8

a. [8p] “A method comprising:”

99. Sutardja-Mathieson performs “[a] method.” The combination switches between high-power and low-power cores. *See* Sutardja ¶¶40, 251.

b. [8a] “monitoring a demand for a multi-core processor by an operating system operating on the multi-core processor,”

100. Sutardja-Mathieson satisfies “monitoring a demand for a multi-core processor by an **operating system** operating on the multi-core processor.” *See* [1f].

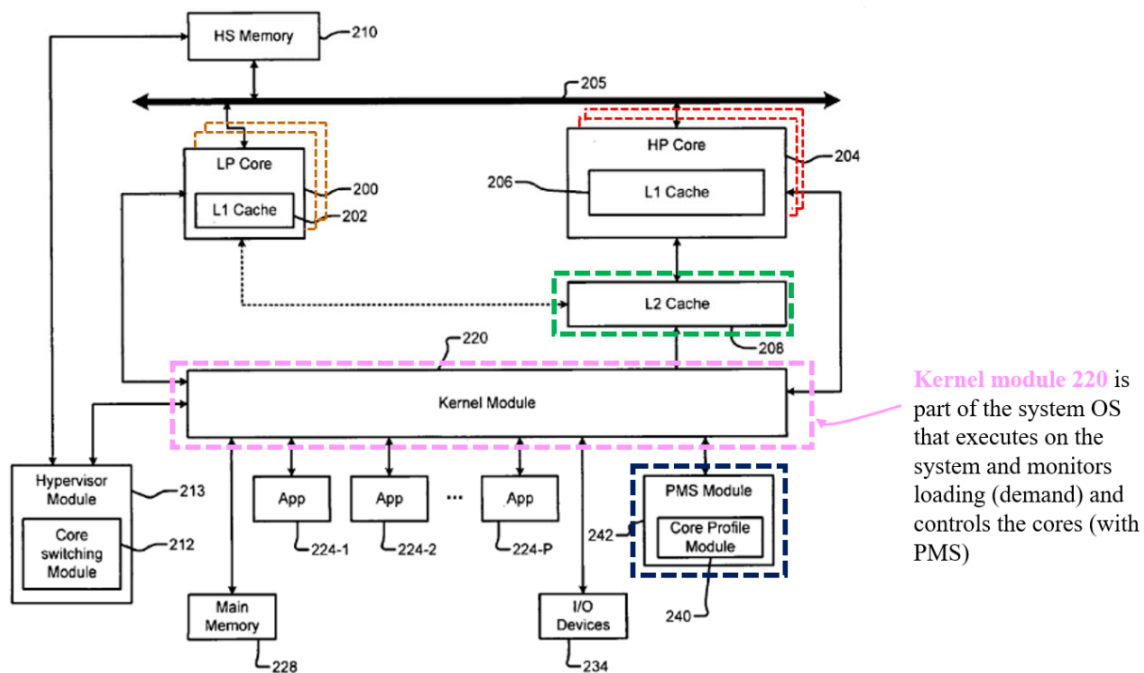


FIG. 5

Sutardja, Fig. 5 (annotated).

- c. **[8b] “wherein the multi-core processor comprises a first plurality of cores and a second plurality of cores that support a same instruction set,”**

101. Sutardja-Mathieson satisfies this feature for the reasons explained in [1a].

- d. **[8c] “the first plurality of cores are higher performance and consume more power than the second plurality of cores,”**

102. Sutardja-Mathieson satisfies this feature for the reasons explained in [1b].

- e. **[8d] “each of the second plurality of cores have a maximum operating frequency that is less than a maximum operating frequency of each of the first plurality of cores,”**

103. Sutardja-Mathieson satisfies this feature for the reasons explained in [1c].

- f. **[8e] “and a caching layer shared by the first plurality of cores and the second plurality of cores; and”**

104. Sutardja-Mathieson satisfies this feature for the reasons explained in [1d].

- g. **[8f] “controlling a core mix of the first plurality of cores and the second plurality of cores based on the demand with power management hardware of the multi-core processor.”**

105. Sutardja-Mathieson satisfies this feature. *See* [1e], [1f].

106. Sutardja explains **PMS module 242** can “initiate or control changes in system operation based on the monitored system loading.” Sutardja ¶251. **PMS module 242** controls a core mix by (i) triggering a **core switching module 212** to enable or disable the **high power** and **low power** cores and (ii) changing operating frequencies based on demand. *Id.* ¶¶252-253.

107. A POSITA would have understood Sutardja’s “monitored system loading” refers to demand. *See id.* ¶251. Sutardja uses the terms “system loading” and “demand” interchangeably, confirming that “**PMS module 242** may control switching between different operating frequencies of the LP core 200 based on demand.” *Id.* ¶252; *see also* Mathieson ¶¶30-31 (controller 240 switching operating modes and thus which cores are enabled “based on the evaluated workload”).

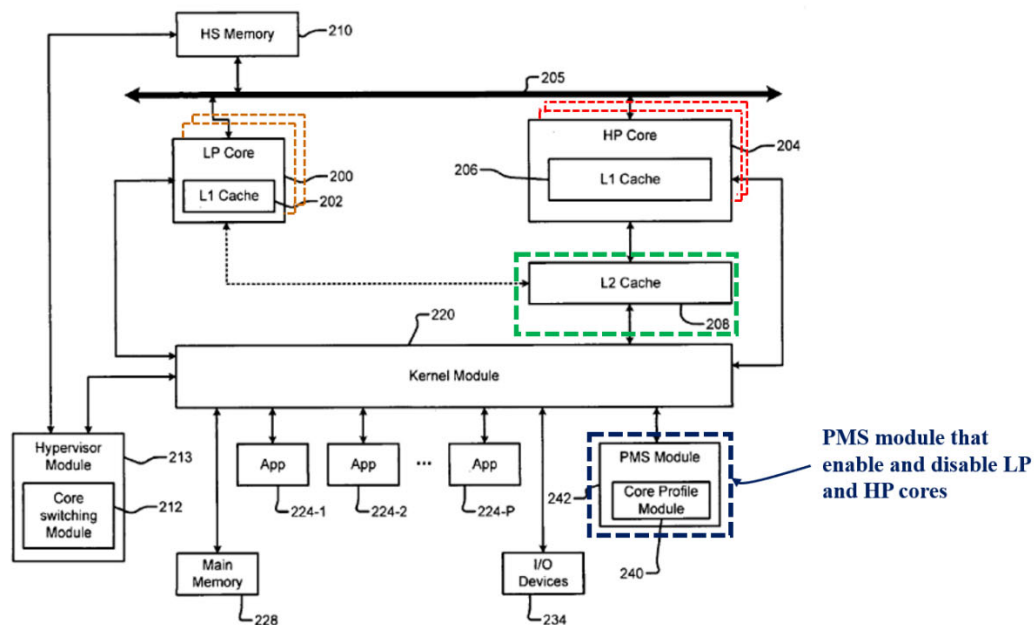


FIG. 5

Id., Fig. 5 (annotated).

7. Claim 15

- a. [15p] “A non-transitory machine readable medium containing code that when processed by a machine causes a method to be performed, the method comprising:”**

108. Sutardja-Mathieson satisfies this feature. *See* Sutardja ¶¶101-129, 172-185, claim 43; Mathieson ¶¶8-10, 20, 23, 79, claim 12.

- b. [15a] “monitoring a demand for a multi-core processor by an operating system executing on the multi-core processor,”**

109. Sutardja-Mathieson satisfies this feature for the reasons explained in [8a], [1f].

- c. [15b] “wherein the multi-core processor comprises a first plurality of cores and a second plurality of cores that support a same instruction set,”**

110. Sutardja-Mathieson satisfies this feature for the reasons explained in [8b], [1a].

- d. [15c] “the first plurality of cores are higher performance and consume more power than the second plurality of cores,”**

111. Sutardja-Mathieson satisfies this feature for the reasons explained in [8c], [1b].

- e. **[15d] “each of the second plurality of cores have a maximum operating frequency that is less than a maximum operating frequency of each of the first plurality of cores,”**

112. Sutardja-Mathieson satisfies this feature for the reasons explained in [8d], [1c].

- f. **[15e] “and a caching layer shared by the first plurality of cores and the second plurality of cores; and”**

113. Sutardja-Mathieson satisfies this feature for the reasons explained in [8e], [1d].

- g. **[15f] “controlling a core mix of the first plurality of cores and the second plurality of cores based on the demand with power management hardware of the multi-core processor.”**

114. Sutardja-Mathieson satisfies this feature for the reasons explained in [8f], [1f].

B. Ground 1B: Sutardja (EX-1002), Mathieson (EX-1003), and Patterson-1 (EX-1004) renders obvious claims 3, 10, and 17

- 1. **Claims 3, 10, 17: “wherein each of the first plurality of cores includes therein a caching layer for data and a caching layer for instructions.”**

115. Sutardja-Mathieson-Patterson-1 satisfies “each of **the first plurality of cores** includes therein **a caching layer for data and a caching layer for instructions.**”

116. As explained above for claim 2, Sutardja explains that **high power cores 204/24** include an **L1 caching layer 206/26** therein. *See* claim 2. A POSITA

would have understood that this caching layer would store data and instructions. *See* Sutardja ¶222. A POSITA also would have found it obvious to fabricate Sutardja's L1 caching layers 206/26 with a caching layer for data and a caching layer for instructions, as disclosed by Patterson-1. Patterson-1, 382 ("Separate caches are found in most recent processors"); *see also* EX-1014 ("Sinharoy") at 4 ("instruction cache (I-cache)" and "data cache (D-cache)"), Fig. 3.

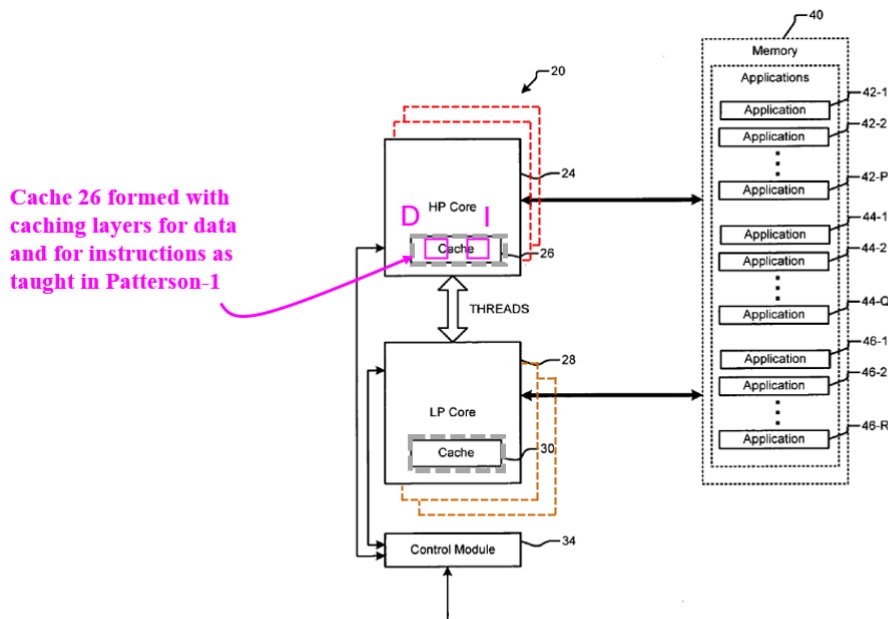
117. Patterson-1 teaches dividing a processor cache into instruction and data caches:

[T]he *data* cache cannot supply all the memory needs of the processor: the processor also needs instructions. Although a single cache could try to supply both, it can be a bottleneck. For example, when a load or store instruction is executed, the pipelined processor will simultaneously request both a data word *and* an instruction word. Hence a single cache would present a structural hazard for loads and stores, leading to stalls. One simple way to conquer this problem is to divide it: one cache is dedicated to instructions and another to data. Separate caches are found in most recent processors

Patterson-1, 382.

118. A POSITA would have been motivated to apply this teaching to Sutardja because Patterson-1 is a textbook that teaches students entering the field of device architecture how to design a memory hierarchy. *See id.*, 372 ("Memory-Hierarchy Design"). A POSITA encountering an architecture like Sutardja's, which is silent as to whether the L1 caching layer is divided, would have understood to

split the **L1 caching layer** into a data caching layer and an instruction caching layer and would have been motivated to do so to speed up data access and reduce bottlenecks, as Patterson-1 teaches. *See* EX-1011 (“Patterson-2”), 552 (“To satisfy the demands of the pipeline without stalling, separate instruction and data caches are used.”), 555 (explaining “many machines use a split instruction and data cache to increase the bandwidth from the cache” to “doubl[e] the cache bandwidth”); EX-1018 (“Arimilli”), 4:41-45 (explaining “those skilled in the art will appreciate that each bifurcated set of L1 caches could alternatively be implemented as a single L1 cache”). The annotated figures below, based on Sutardja’s Figures 3A and 5, represent the Sutardja-Mathieson-Patterson-1 combination.

**FIG. 3A**

Sutardja, Fig. 3A (annotated).

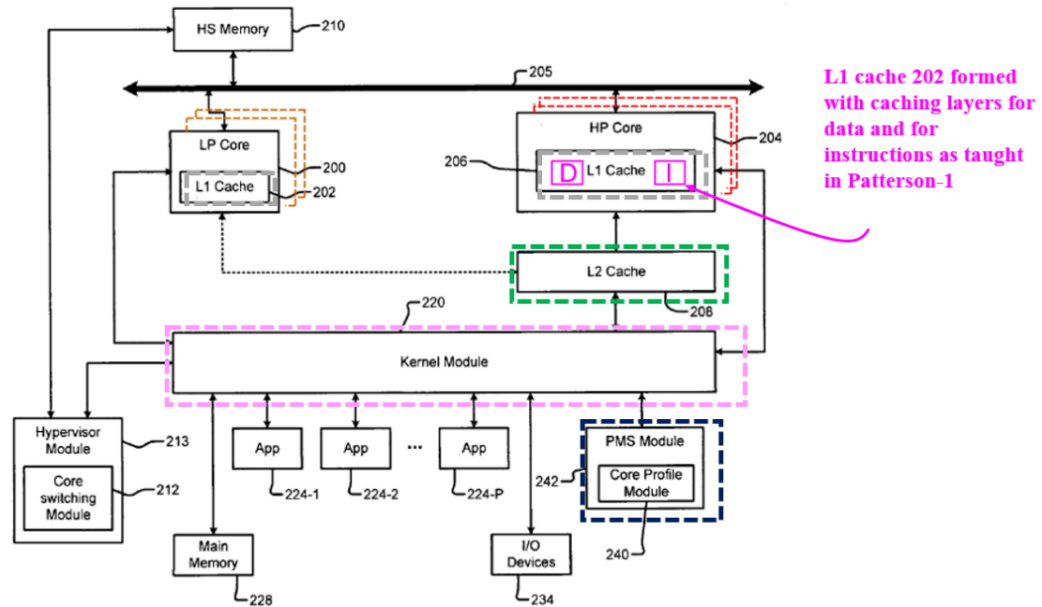


FIG. 5

Id., Fig. 5 (annotated).

119. Combining Patterson-1's divided cache into the Sutardja-Mathieson system would have involved nothing more than combining known prior-art elements (Sutardja's processing system and Patterson-1's split L1 caches) according to known methods (forming L1 cache layers with instruction and data layers) to yield predictable results (caches with improved speed/bandwidth and reduced bottlenecks). It also would have involved nothing more than using one known technique (Patterson-1's split L1 caches) to improve similar devices (Sutardja's L1 caches) in the same way (by including split cache layers for data and instructions). A POSITA would have had a reasonable expectation of success in the combination.

C. Ground 1C: Sutardja (EX-1002), Mathieson (EX-1003), and Knebel (EX-1005) renders obvious claims 5, 12, and 19

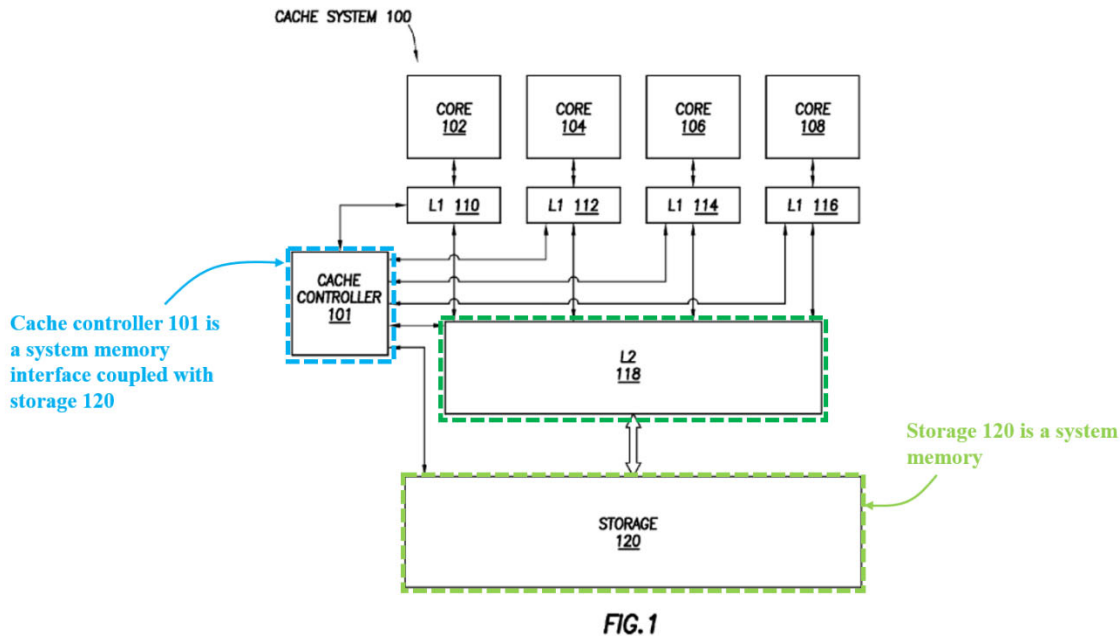
1. **Claim 5, 12, 19: “a system memory interface to couple to a system memory to search for a cache line, requested by any core of the first plurality of cores and the second plurality of cores, that is not found in the caching layer” / “coupling a system memory interface of the multi-core processor to a system memory to search for a cache line, requested by any core of the first plurality of cores and the second plurality of cores, that is not found in the caching layer”**

120. A POSITA would have understood that it was conventional and understood that processor systems search a cache and then search memory if the data is not in the cache. Sutardja-Mathieson-Knebel also expressly discloses this. For example, Sutardja-Mathieson-Knebel includes “a system memory interface to couple to a system memory to search for a cache line, requested by any core of the first plurality of cores and the second plurality of cores, that is not found in the caching layer.”

121. Sutardja discloses a high-speed memory 210 integrated with the cores. Sutardja ¶247, Fig. 5. Memory 210 is connected through a bus 205 that allows communication between memory 210 and the cores *Id.* A POSITA would have found it obvious to include such an interface in Sutardja’s architecture.

122. Knebel, for example, teaches the same cache architecture as Sutardja—multiple cores with individual L1 caches and a shared L2 cache connected to a memory. EX-1005 (“Knebel”) ¶¶10-12, Fig. 1. Knebel further describes a cache

controller 101 that searches the caches. *Id.*, ¶20. Figure 1 illustrates Knebel's **cache controller 101**, **L2 cache**, and system memory **storage 120**.



Id., Fig. 1 (annotated).

123. A POSITA would have recognized that a cache controller is necessary to search Sutardja's caches and thus would have incorporated a cache controller like Knebel's **cache controller 101** in Sutardja's architecture. Applying Knebel's teachings to Sutardja, a POSITA would have found it obvious for Sutardja's **high-speed memory 210** to "search for a cache line, requested by any core of the first plurality of cores and the second plurality of cores, that is not found in the caching layer," as claimed.

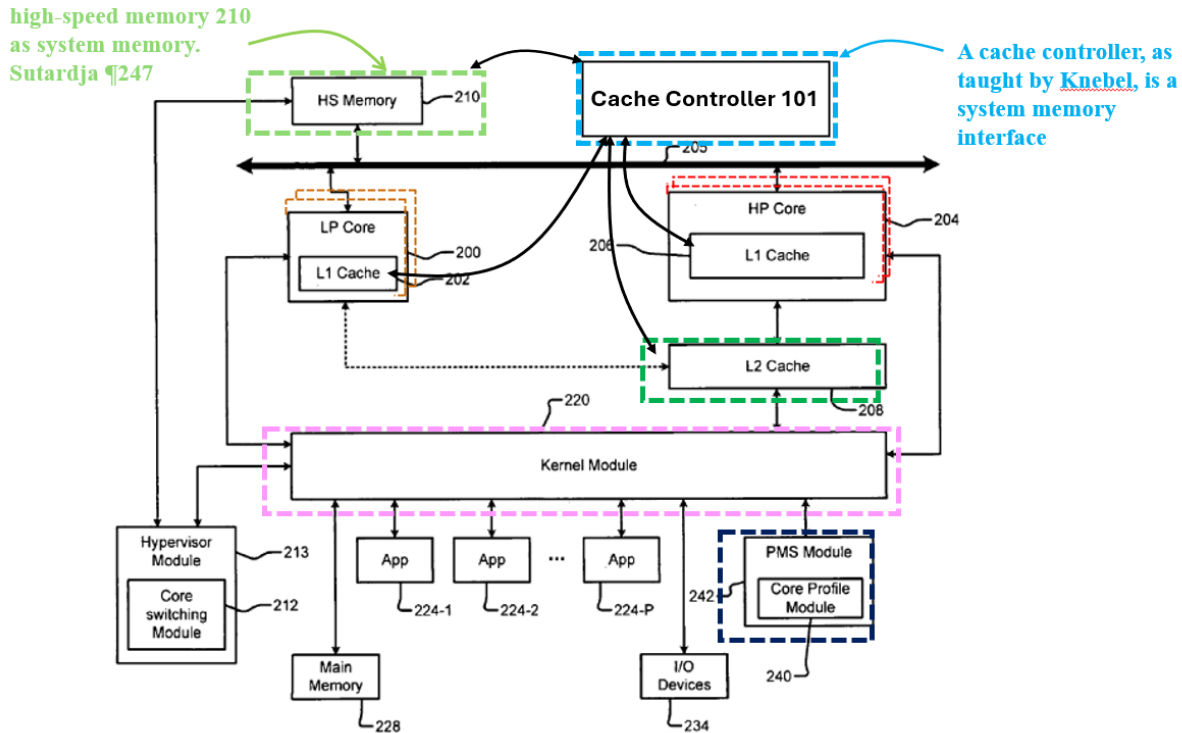


FIG. 5

Sutardja, Fig. 5 (annotated and altered above).

124. Knebel teaches that, when a core requires data stored in a particular address, it first determines whether the data is in the lowest level cache (here, the L1 cache). Knebel ¶12. If the data is not in the L1 cache, the core next determines whether the data is in the next-level cache (here, the shared **L2 cache**). *Id.* If the data is not found there either, “the core obtains the data from the **storage 120**.” *Id.*

125. Knebel describes a fundamental, textbook operation of an interface between caches and system memory. Applied to Sutardja’s architecture, a POSITA would have understood that, when cache line requests sent by the **high power** and **low power** cores are not found in the cache layers (either the **L2 cache** or L1

caches), **memory 20** would be searched for the cache line. When a request from cores is not found in cache layers, the event is known as a “cache miss.” See Patterson-2, 545; EX-1009 (“Censier & Feautrier”), 1112. The textbook way to resolve a “cache miss” is for the memory controller to search the memory for the missing cache information. Patterson-2, 550-51.

126. Patterson-2, which teaches techniques for handling cache misses, explains that in multi-core processors like Sutardja’ that “[t]he control unit must detect a miss and process the miss by fetching the data from memory (or a lower-level cache).” *Id.*, 550. To do this, Patterson-2 explains “[t]he basic approach is to stall the CPU, freezing the contents of all the registers. A *separate controller* handles the cache miss, fetching the data into the cache from memory. Once the data is present, execution is restarted at the cycle that caused the cache miss.” *Id.*, 551 (emphasis added).

127. A POSITA would have been motivated to apply Knebel’s techniques for handling cache misses to Sutardja’s architecture because, in addition to Knebel’s own teachings, Patterson-2’s textbook teachings specifically disclose to students entering the field of device architecture that cache misses should be handled by a controller to fetch the data from a memory (as Knebel teaches). See Patterson-2, 550. A POSITA encountering a cache system like Sutardja’s thus would have applied

Knebel's teachings such that a system memory interface in Sutardja would search for the cache line in memory in the event of a cache miss.

128. The Sutardja-Mathieson-Knebel combination would have involved nothing more than combining prior art elements (understanding Sutardja's memory 20 would include a memory interface that applies Knebel's system memory interface and technique for handling cache misses in processing systems like Sutardja's) according to known methods (fetching data from memory in the event of a cache miss) to yield predictable results (a processor that handles cache misses by fetching the data from memory). It also would have involved no more than using a known technique (Knebel's system memory interface and technique for handling cache misses in a processing system) to improve similar devices (Sutardja's multi-core processor) in the same way (understanding Sutardja's memory 20 would include a memory interface to search it in the event of a cache miss to improve the reliability of accessing information). The teaching and suggestion in the prior art (e.g., Patterson-1's teaching that cache misses in processors like Sutardja's should be handled by fetching the data from memory, as described by Knebel) also would have led a POSITA to combine Sutardja, Mathieson, and Knebel. A POSITA would have had a reasonable expectation of success in the combination.

D. Ground 1D: Sutardja (EX-1002), Mathieson (EX-1003), and Liu (EX-1006) renders obvious claims 7, 14, and 21

1. Claims 7, 14, 21: “coherency logic circuitry”

129. Sutardja-Mathieson-Liu satisfies “**coherency logic circuitry** to maintain cache coherency in **the caching layer** shared by **the first plurality of cores** and **the second plurality of cores**.”

130. When multiple cores share memory, as in Sutardja, maintaining cache coherency is fundamental. *See* EX-1013 (“Tucker”), 19-11 (“[I]t is necessary to maintain coherency among the caches such that any memory access is guaranteed to return the latest version of the data in the system.” (citing EX-1009, Censier & Feautrier)); Arimilli, 1:42-46 (explaining “it is important to maintain a coherent memory hierarchy”); EX-1010 (“Akhiani”), 1 (“The cache-coherence protocol is a fundamental component of any shared-memory multiprocessor design”); EX-1009, (Censier & Feautrier), Abstract.

131. Without maintaining cache coherency, the caches can hold inconsistent values for the same data or instruction read from, or to be written into, a given memory address. *See, e.g.*, EX-1009, (Censier & Feautrier), 1113 (“A memory scheme is coherent if the value returned on a LOAD instruction is always the value given by the latest STORE instruction with the same address.”). To perform this fundamental task, coherency logic is necessary. Because Sutardja-Mathieson includes a multiprocessor architecture with a shared memory, it must maintain cache

coherency in all the caches. Tucker at 19-11; Arimilli, 1:42-46; Akhiani, 1; Censier & Feautrier, Abstract. Being so critical, it would have been obvious.

132. To perform this fundamental task, a POSITA would have found it obvious to use cache coherency logic circuitry. For example, a POSITA would have recognized that a typical snooping protocol would maintain cache coherence in the L1 caches. *See, e.g.*, EX-1012 (“Herrero”), 1 (“Cache coherence in multiprocessors has been traditionally granted either by snoop or directory based schemes.”). In a typical snooping protocol, “an external agent or core attempts to retrieve data from a cache, such as a core cache, by first checking (‘snooping’) to see if the data resides in a particular cache.” EX-1006 (“Liu”) ¶3. To maintain coherence between a shared cache and the core caches, “cache line information is maintained for each line of a shared inclusive cache.” *Id.* ¶5. To do so, “accesses must be made to the core caches.” *Id.* ¶4.

133. It was well known to POSITA that a shared cache acts as a filter on the coherence traffic. As the core caches are repeatedly snooped, traffic accumulates on internal buses. *See id.* ¶3. A POSITA would have been motivated to reduce such traffic, because that is one of the main reasons to use a shared cache in the first place. *See id.* ¶4 (explaining “[t]he shared cache is a prior art attempt to reduce the traffic on internal buses”). A shared cache can filter the snoop request to the core caches since any data that is not in shared cache can avoid snooping the core caches. Thus,

a POSITA would have been motivated to apply the teachings of Liu and include a snoop filter in the shared **L2 cache**. As Liu explains, a snoop filter “can reduce the snoop traffic seen by each processor core.” Liu ¶18. A POSITA would have applied Liu’s use of a snoop filter in Sutardja-Mathieson to maintain cache coherency while reducing traffic. A POSITA further would have understood a snoop filter is “coherency logic circuitry to maintain cache coherency” that would be located in the shared caching layer. *See* Liu ¶4 (explaining a shared cache acts as a “filter” cache for the core caches, and operations “must take place between the shared and core caches to ensure shared cache coherency”).

134. The Sutardja-Mathieson-Liu combination would have involved nothing more than combining prior art elements (e.g., Liu’s use of coherency logic circuitry to maintain coherency in Sutardja’s caching architecture) according to known methods (snoop coherency circuitry) to yield predictable results (an operational multiprocessor device with reduced traffic to the core caches). This would have involved the use of known technique (using logic circuitry, as in Liu, to maintain cache coherency in a shared-memory multiprocessor) to improve similar devices (Sutardja’s shared-memory multiprocessor) in the same way (e.g., snoop coherency logic circuitry, as in Liu). Liu’s teaching, suggestion, and motivation also would have led a POSITA to modify Sutardja-Mathieson to arrive at the claimed invention.

E. Ground 2A: Mathieson (EX-1003) and Sakarda (EX-1007) renders obvious claims 1-4, 6, 8-11, 13, 15-18, and 20

1. Motivation to Combine Mathieson and Sakarda

135. A POSITA would have found it obvious to combine Mathieson and Sakarda. In a first combination, a POSITA would have found it obvious to incorporate the following teachings from Sakarda into Mathieson's architecture: (1) cores that execute the same computer instructions; and (2) operating slow cores in a lower frequency range than fast cores. In a second, alternative combination, a POSITA would have found it obvious to incorporate Mathieson's disclosure of multiple, low-power cores into Sakarda's architecture.

First Mathieson-Sakarda Combination

136. Mathieson teaches a CPU 102 with a first set of at least one core 220 and a second set of at least one core 210. Mathieson, Fig. 2, ¶28. Mathieson suggests that its cores execute the same computer instructions by the cores having "substantially the same functionality," *id.* ¶26, and selecting "which set of cores should execute *the* processing operations" based on "supply voltage, a temperature of each chip included in the processing complex, and an average leakage current over a period of time of each chip." *Id.* ¶39. However, Mathieson does not expressly state that its cores execute the same computer instructions.

137. A POSITA would have found it obvious to incorporate Sakarda's teaching to implement cores that execute the same computer instructions into

Mathieson's architecture. EX-1007 ("Sakarda"), 1:44-49 (the first and second processor cores executing "the computer instructions"), 4:17-21 (determining when "to switch execution of the computer instructions 110 from one processor core to another."), 8:57-64 ("execution of the computer instructions 110 is continued on processor core 104.").

138. The prior art confirms that a POSITA would have known to implement cores that use the same instructions sets to create cores with the same functionality. *See, e.g.*, EX-1015 ("Fedorova"), 1 ("Using the same ISA on all cores means that you can run the same binary on all cores and not have to compile your code with a different compiler for each core type."). A POSITA thus would have understood and recognized the advantages of applying Sakarda's teaching of cores that support the same instruction set when implementing Mathieson's architecture.

139. A POSITA would have been motivated to combine Sakarda with Mathieson in this way because both address similar problems in the art, including conserving power in multicore processors with both fast cores and slow cores. *See* Mathieson ¶20; Sakarda at 1:44-49. Such a combination would have involved nothing more than combining known prior art elements (e.g., Mathieson's multi-core processor with cores that have substantially the same functionality and Sakarda's use of cores that execute the same computer instructions) according to known methods (e.g., instantiating processor cores that support the same instruction

set) to yield predictable results (e.g., a more efficient processor with multiple cores that can execute the same instructions to implement the same functionality). Such combination would have resulted in a system with easier workload sharing and more efficient core usage, also facilitating switching between cores with substantially the same functionality. A POSITA would have had a reasonable expectation of success in this combination.

140. A POSITA also would have found it obvious to incorporate a second aspect of Sakarda into Mathieson's architecture: slow cores operating with a lower "maximum operating frequency" than fast cores. Like Mathieson, Sakarda teaches multi-core processors that operate with different performance specifications (such as processing rates) and power characteristics. Sakarda at 3:19-25. For example, Sakarda's slow core 104 operates at a processing rate between 30 MHz and 100 MHz, whereas fast core 102 operates between 100 MHz and 800 MHz. *Id.*, 3:25-30. According to Sakarda, using these different frequency ranges reduces power consumption. *Id.*, 3:30-32. Sakarda thus teaches an architecture in which a second core 104 operates within a lower frequency range than a first core 102. Mathieson likewise employs fast cores and slow cores to conserve power. Mathieson, Abstract ("[T]o achieve the lowest total power consumption.").

141. In view of these known teachings, a POSITA would have been motivated to employ Sakarda's teaching to use different frequency ranges for two

sets of cores in Mathieson's architecture. For example, a POSITA would have applied Sakarda's teachings to operate Mathieson's slow cores in a frequency range below Mathieson's operating frequency threshold 610, and to operate Mathieson's fast cores in a frequency range above operating frequency threshold 610. Mathieson ¶74, Fig. 6. In this implementation, Mathieson's cores would minimize power consumption. Indeed, that is precisely what Sakarda teaches, with the exemplary operating frequency threshold being 100 MHz. Sakarda, 3:25-30.

142. A POSITA would have understood the Sakarda-Mathieson combination would have involved nothing more than combining known elements (e.g., Mathieson's multi-core processor, including fast cores and slow cores, and Sakarda's technique of only operating the slow cores at a lower frequency range than the fast cores) according to known methods (e.g., clocking the slow and fast cores at different rates) to yield predictable results (e.g., a multicore processor that conserves power by operating the slow cores and fast cores at appropriate frequencies). Such combination would have reduced overall power consumption. A POSITA would have had a reasonable expectation of success in the combination.

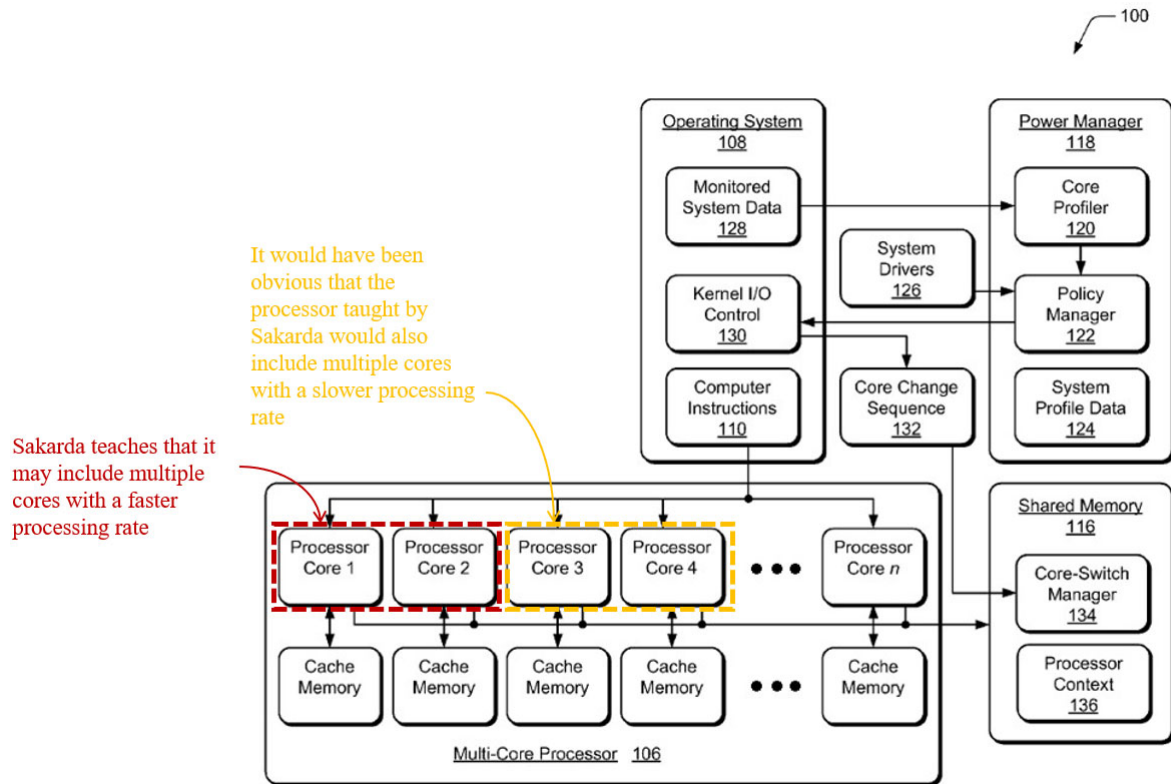
Second Mathieson-Sakarda Combination

143. Additionally, a POSITA would have found it obvious to modify Sakarda's architecture, which employs multiple high-power cores and at least one low-power core, by applying Mathieson's teaching to use multiple low-power cores.

See Mathieson ¶¶25, 28. For example, Mathieson explains that multiple “slow” cores enable processing of complex operations with less power, allow simpler switching of tasks from high to low power cores, and improve parallelism and performance. *Id.* ¶¶29, 34, 57.

144. The prior art confirms this would have been obvious. For example, Carmack explains that adding more low-power cores would be matter of design choice in designing processors with improved maximum capabilities. Carmack ¶28 (describing the tradeoff and benefits in the “choice of the number of cores”). A POSITA thus would have recognized that applying Mathieson’s teaching to use multiple, low-power cores would produce similar advantages as the ones Mathieson describes when applied to Sakarda’s architecture.

145. The figure below, adapted from Sakarda’s Figure 1, illustrates the Mathieson-Sakarda architecture when implemented with multiple slow cores and multiple fast cores:



Sakarda, Fig. 1 (annotated and altered above).

146. A POSITA would have recognized that this Mathieson-Sakarda combination would have involved nothing more than combining known elements (e.g., Sakarda's multi-processor architecture and Mathieson's teaching to use multiple low-power cores) according to known methods (e.g., instantiating multiple low-power cores instead of just one) to yield predictable results (e.g., a chip with multiple low-power cores). This would have produced desirable results by increasing the processing capacity at lower power, increasing parallelism, and optimizing performance and power consumption. See Mathieson ¶¶33-35 (explaining how switching between slow and fast cores based on the workload and

performance demands optimizes performance), ¶57 (explaining “less power would be consumed” and describing advantages of switching between cores). A POSITA would have had a reasonable expectation of success in the combination.

2. Claim 1

a. [1p] “A multi-core processor comprising:”

147. The Mathieson-Sakarda combinations include “[a] multi-core processor.” The first combination includes a CPU 102 with a **first set of at least one core 220** and a **second set of at least one core 210**. See Mathieson, Fig. 2, ¶28.

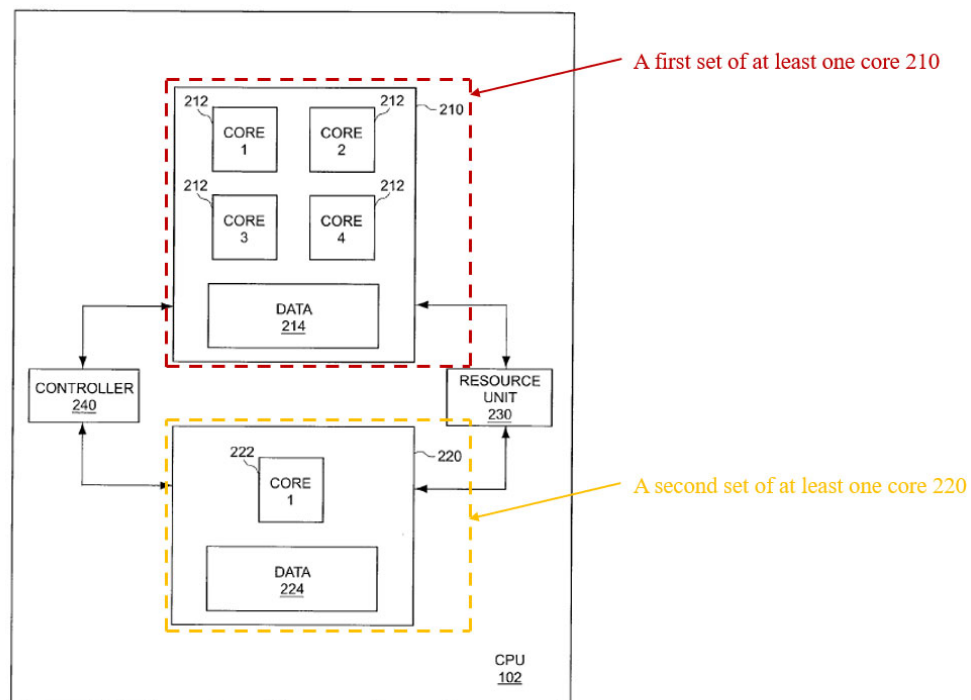
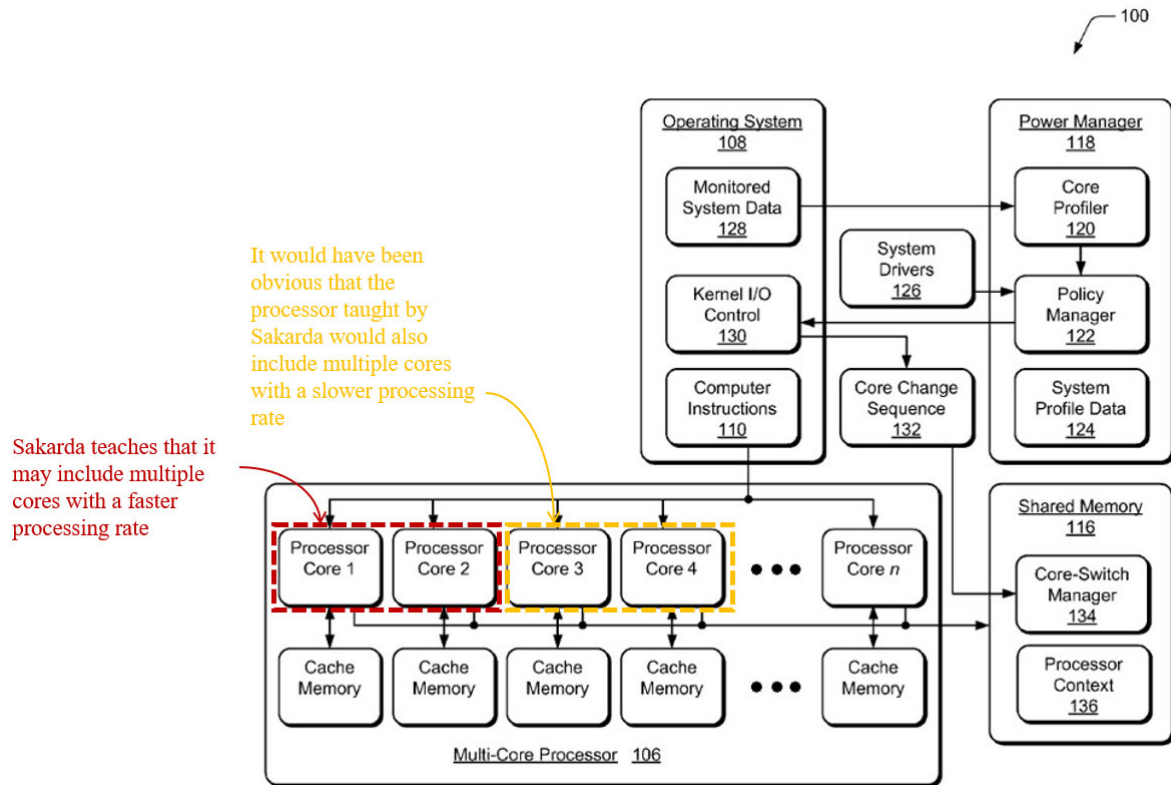


FIG. 2

Id., Fig. 2 (annotated).

148. In the second combination, Sakarda teaches a multi-core processor system 100. Sakarda, 2:27-29, 3:2-45, Fig. 1. This includes n cores, each with at least a first processor core 102 and a second processor core 104 (up to n cores). *Id.*



Id., Fig. 1 (annotated and altered above).

- b. [1a] “a first plurality of cores and a second plurality of cores that support a same instruction set,”

149. Both Mathieson-Sakarda combinations include “a **first plurality of cores** and a **second plurality of cores** that support a **same instruction set**.”

150. The first combination includes a **first set of cores 210** and **second set of cores 220**. See Mathieson, Fig. 2, ¶25. Mathieson explains each set of cores may include multiple cores: “N” in the **first set of cores 210** and “M” in the **second set**

of cores 220. *Id.* ¶28 (“According to various embodiments, the first set of cores includes N cores and the second set of cores includes M cores. In one embodiment, N is not equal to M. In other embodiments, N is equal to M.”).

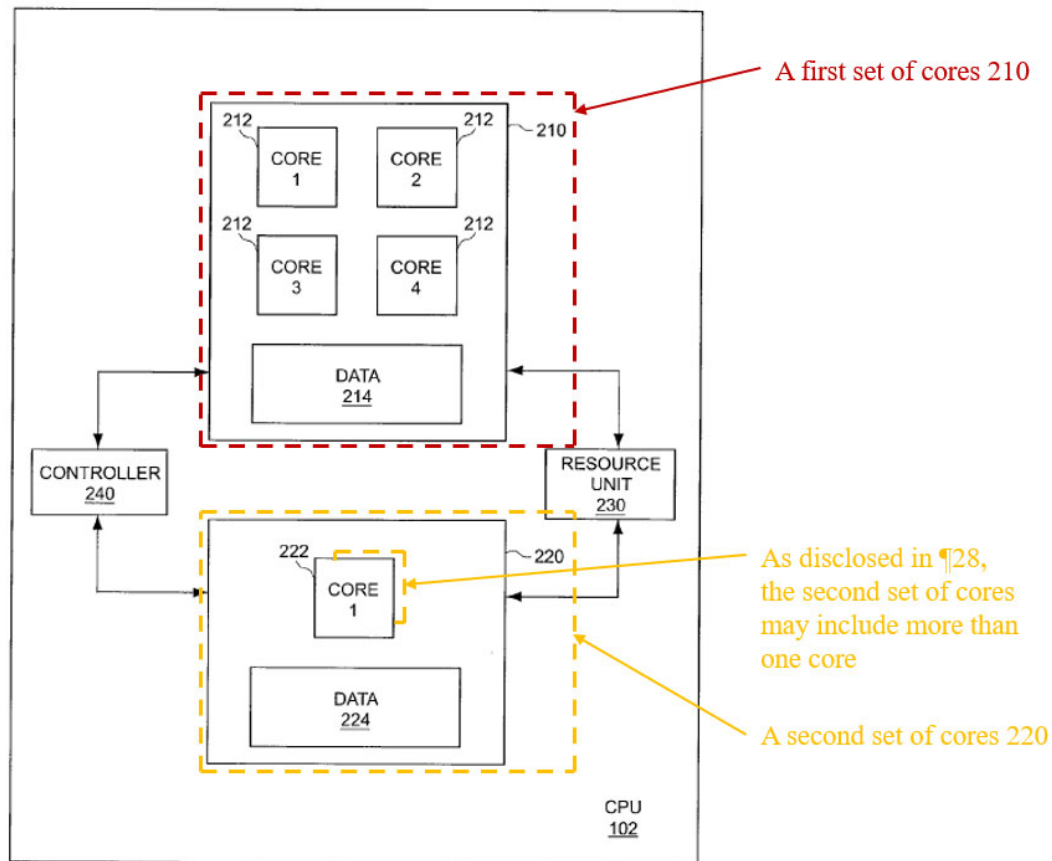


FIG. 2

Id., Fig. 2 (annotated).

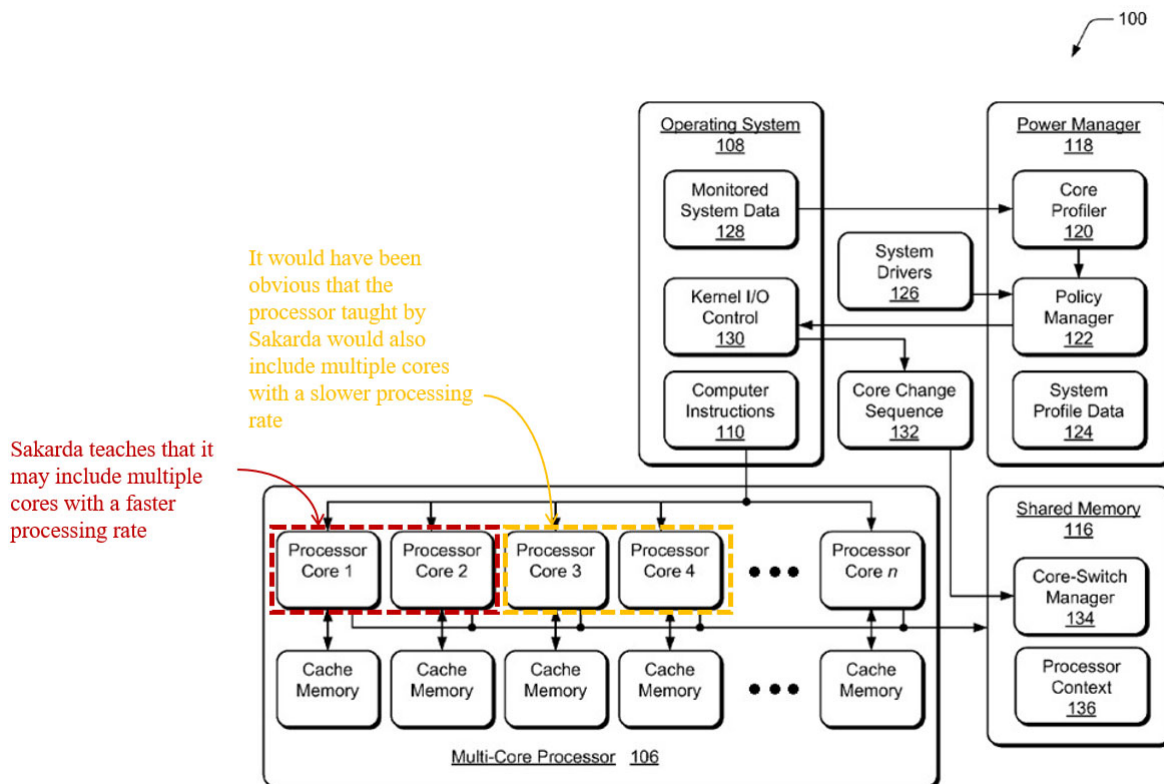
151. Mathieson suggests the **first set of cores 210** and the **second set of cores 220** support a same instruction set since the two sets of cores “may implement substantially the same functionality.” *Id.* ¶26. The first combination incorporates Sakarda’s teachings, such that Mathieson’s cores execute the same instruction set.

See supra §VIII.E.1 (“First Mathieson-Sakarda Combination”); Sakarda, 1:44-49, 4:17-21, 8:57-64, Fig. 1.

152. In the second Mathieson-Sakarda combination, the system includes a first plurality of cores and a second plurality of cores. *See supra* § VIII.E.1 (“Second Mathieson-Sakarda Combination”). According to Sakarda, those cores can be implemented “with different performance and/or power characteristics.” Sakarda, 3:19-25. Additionally, Sakarda provides an example of a multi-core processor that includes “three processor cores, such as **two processor cores** that have a faster processing rate and a **third processor core** that has a slower processor rate.” *Id.*, 4:37-40. A POSITA would have understood the two processor cores that have a faster processing rate are a **first plurality of cores** and would have found it obvious that the third processor core is part of a **second plurality of cores** with a slower processing rate. *See id.*; *supra* §VIII.E.1 (“Second Mathieson-Sakarda Combination”).

153. Sakarda’s system 100 includes n cores, Sakarda, 3:3-15, which a POSITA would have understood to be any number of cores, including more than three. A POSITA would have desired two or more of Sakarda’s cores to have the slower processor rate. For example, Sakarda explains that multi-core processor system 106 includes “at least” one **processor core** “that executes the operating system instructions at a second processing rate that is different than the first

processing rate.” *Id.*, 7:15-18. A POSITA thus would have understood Sakarda’s multi-core processor may include more than one core that executes instructions at a second processing rate. *See supra* § VIII.E.1 (“Second Mathieson-Sakarda Combination”); Mathieson ¶¶33-35, 57. The figure below, adapted from Sakarda’s Figure 1, represents the second combination:



Id., Fig. 1 (altered and annotated).

- c. [1b] “wherein the first plurality of cores are higher performance and consume more power than the second plurality of cores, and”

154. Mathieson-Sakarda combinations satisfy “wherein the **first plurality of cores** are higher performance and consume more power than the **second plurality of cores**.”

155. The first combination includes a **set of “fast” cores 210** and a **set of “slow” cores 220**. Mathieson ¶28. The **fast cores** are “associated with higher performance” than the **slow cores**. *Id.* ¶20.

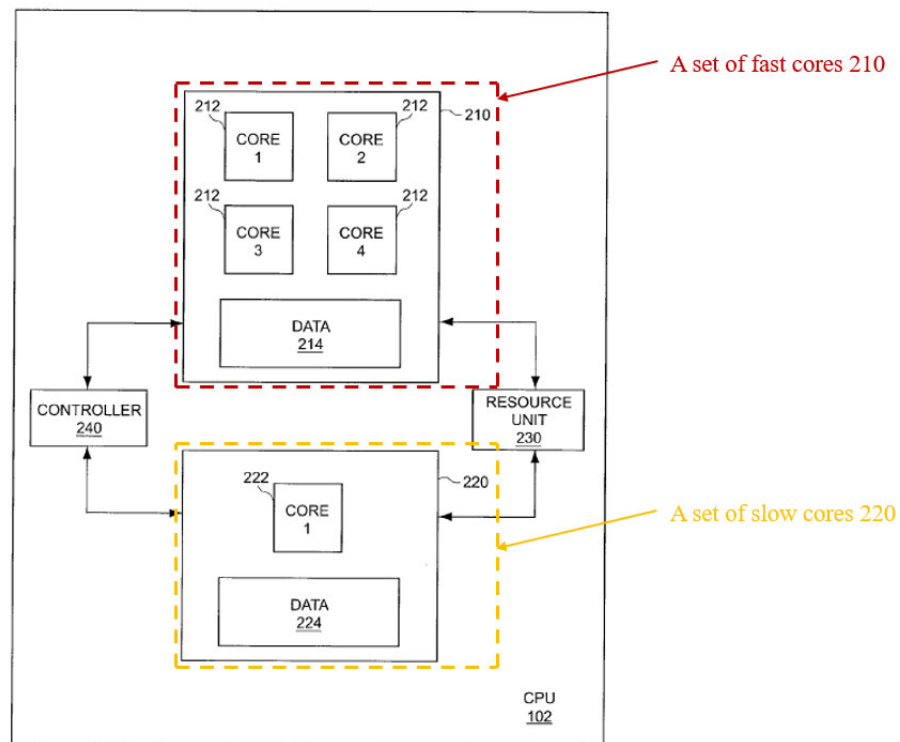


FIG. 2

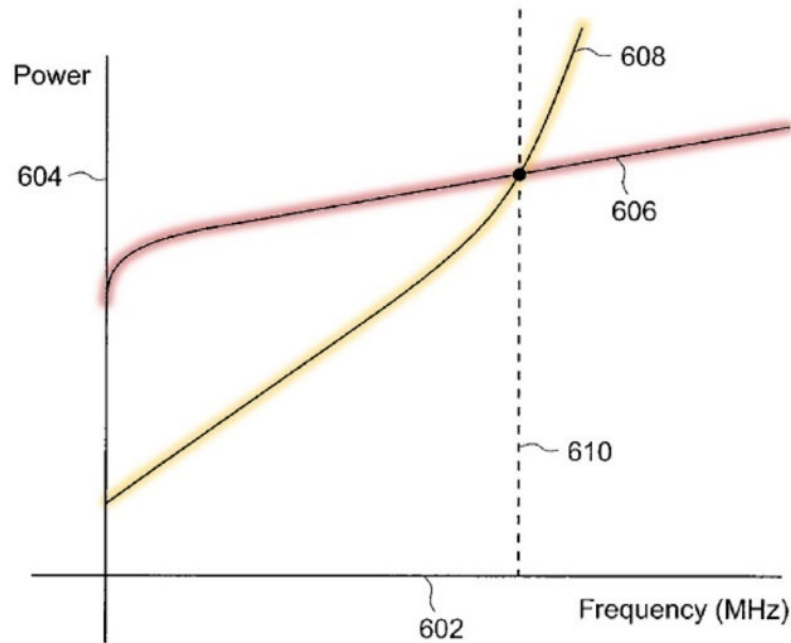
Id., Fig. 2 (annotated).

156. Mathieson explains the **fast cores** consume more power than the **slow cores**, as claimed. Specifically, the frequency of the **slow cores** is controlled to be less than the frequency of the **fast cores**, such that **slow cores** consume less power than **fast cores**. *Id.*, Fig. 6, ¶¶72-75 (“[W]hen operating the processing complex at lower frequencies, executing the processing operations with the slow cores is associated with lower total power consumption”), ¶¶29-32.

157. Mathieson further explains the **fast cores** are associated with “higher leakage power” than the **slow cores**. *Id.* ¶20 (“[T]he cores 130 are associated with higher performance and higher leakage power than the cores 140”). A POSITA would have understood that, because the **fast cores** have a higher leakage power, their power consumption tends to increase. *Id.* ¶73 (“[T]he lower total power associated with operating the processing complex at lower frequencies using the **slow cores** is based on the lower leakage power associated with the **slow cores**.”).

158. In Figure 6, Mathieson depicts a transition frequency 610 above which slow cores consume more power than fast cores. *Id.*, Fig. 6 (annotated below), ¶74. Mathieson teaches to avoid such a scenario, explaining CPU 102 will switch from using fast cores to using slow cores when doing so would use less power. *Id.* ¶¶31-32 (“In the second mode of operation, the second set of [slow cores] is enabled and operable and the first set of [fast cores] is disabled.”), ¶75 (“[A] controller included in the processing complex determines whether executing the processing operations

with the fast cores or executing the processing operations with the slow cores achieves lower power consumption.”).



Id., Fig. 6 (annotated).

159. “[T]he determination of which type of cores to use when executing the processing operations may be based on operating frequency, as shown in Fig. 6.” *Id.* ¶75; *see also id.* ¶32 (explaining “the processing parameter may be a processing frequency”). As Figure 6 demonstrates, **slow cores** consume less power than the **fast cores** below threshold 610. *Id.* ¶74. Based on Mathieson’s CPU configuration, a POSITA would have understood Mathieson’s **slow cores** operate at frequencies that consume less power than **fast cores**. A POSITA thus would have understood that Mathieson’s **slow cores** consume less power than its **fast cores**. *See* Mathieson ¶¶31, 73-75, Fig. 6.

160. A POSITA would have found it obvious—whether based on these teachings in Mathieson teachings alone or in combination with Sakarda—to operate Mathieson’s **slow cores** at a lower power than its **fast cores** in the first Mathieson-Sakarda combination.

161. In the second combination, Sakarda’s multi-core processor system 100 “can be implemented with different performance and/or power characteristics.” Sakarda, 3:19-21; *see also id.*, 3:3-12, Fig. 1. For example, one **processor core** executes computer instructions 110 “at a first processing rate,” and another **processor core** executes instructions 110 “at a second processing rate that is different than the first processing rate.” *Id.*, 3:21-25. In one example, Sakarda discloses “three processor cores, such as **two processor cores that have a faster processing rate** and **a third processor core that has a slower processor rate.**” *Id.*, 4:37-40. A POSITA would have found it obvious to implement two or more such low-power cores in Sakarda’s architecture, as Mathieson teaches. *See; supra* §VIII.E.1 (“Second Mathieson-Sakarda Combination”); Mathieson ¶¶25, 28, 33-35, 57.

162. Sakarda teaches n cores, including more than three. Sakarda, 3:2-17, Fig. 1. A POSITA would have implemented **at least** one of these cores at a slower processor rate. *See* Sakarda, 7:15-18 (explaining “multi-core processor 106 includes **at least** the second processor core 104 that executes the operating system instructions

at a second processing rate that is different than the first processing rate”). Since the processor includes “at least” one core that executes the instructions at a second processing rate, it may include more than one core that executes the instructions at a second processing rate. *See supra* § VIII.E.1 (“Second Mathieson-Sakarda Combination”). For example, Mathieson teaches to use multiple **slow cores** (a **second set of cores 220**). According to Mathieson, that **second set of cores 220** may process operations using less power, improve parallelism, and optimize performance and power consumption. *See* Mathieson ¶¶33-35, 57.

- d. [1c] “each of the second plurality of cores have a maximum operating frequency that is less than a maximum operating frequency of each of the first plurality of cores;”

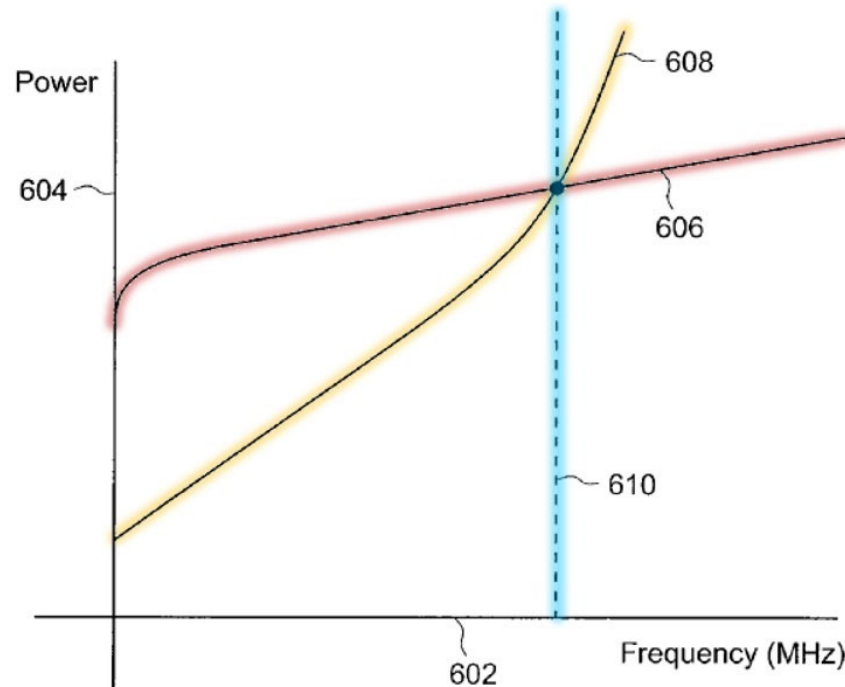
163. In both Mathieson-Sakarda combinations, “each of the **second plurality of cores** have a **maximum operating frequency** that is less than a **maximum operating frequency** of each of the **first plurality of cores**.”

164. Mathieson explains that the **first set of cores** and the **second set of cores** each has an operating frequency range. Mathieson ¶34, claim 4. According to Mathieson, the **slow cores** are “constructed from slower transistors that are not capable of operating as quickly as the transistors includes in the **cores 212**.” *Id.* ¶29. Accordingly, a POSITA would have understood the **slow cores** have a lower **maximum operating frequency** than the **maximum operating frequency** of the **fast cores**.

165. A POSITA also would have found it obvious to operate Mathieson's **slow cores** at a lower maximum operating frequency than its **fast cores** based on Mathieson's teachings (*see id.* ¶74, Fig. 6) and/or Sakarda's teachings. In Sakarda's architecture, a **second core** operates within a lower frequency range (between 30 MHz and 100 MHz) than a **first core** (between 100 MHz and 800 MHz). Sakarda, 3:25-30. The 100 MHz maximum operating frequency of Sakarda's **slow cores** is less than the 800 MHz maximum operating frequency of Sakarda's **fast cores**. Sakarda therefore teaches a second plurality of cores having a maximum operating frequency that is less than a maximum operating frequency of a first plurality of cores. According to Sakarda, using these different frequency ranges reduces power consumption. *Id.*, 3:30-32 (conserving power when computer instructions 110 "are executed at the slower processing rate on processor core 104."), 2:53-55 (same). This accords with Mathieson's teachings, which likewise employs **fast cores** and **slow cores** to conserve power. *See, e.g.*, Mathieson, Abstract ("achieve the lowest total power consumption").

166. As explained above, Mathieson's architecture consumes less power when the **slow cores** are operated in a lower frequency range than the **fast cores**, with a **maximum frequency 610**. *See supra* [1a], [1b]; Mathieson ¶¶31, 73-75, Fig. 6. Accordingly, a POSITA would have been motivated to operate Mathieson's slow cores at a lower maximum frequency than its fast cores to conserve power. For

example, a POSITA would have found it obvious to operate **low-power cores (path 608)** at a lower frequency than the **high-power cores (path 606)**—namely, below an “operating frequency threshold 610”—in both combinations. Mathieson, ¶74, Fig. 6; *see also* Sakarda at 3:25-30.



Mathieson, Fig. 6 (annotated).

e. [1d] “a caching layer shared by the first plurality of cores and the second plurality of cores; and”

167. Both Mathieson-Sakarda combinations include “a **caching layer** shared by the **first plurality of cores** and the **second plurality of cores**.” In the first combination, Mathieson’s **first set of cores** and **second set of cores** may share an **L2 cache 230**. *See* Mathieson ¶45, Fig. 2.

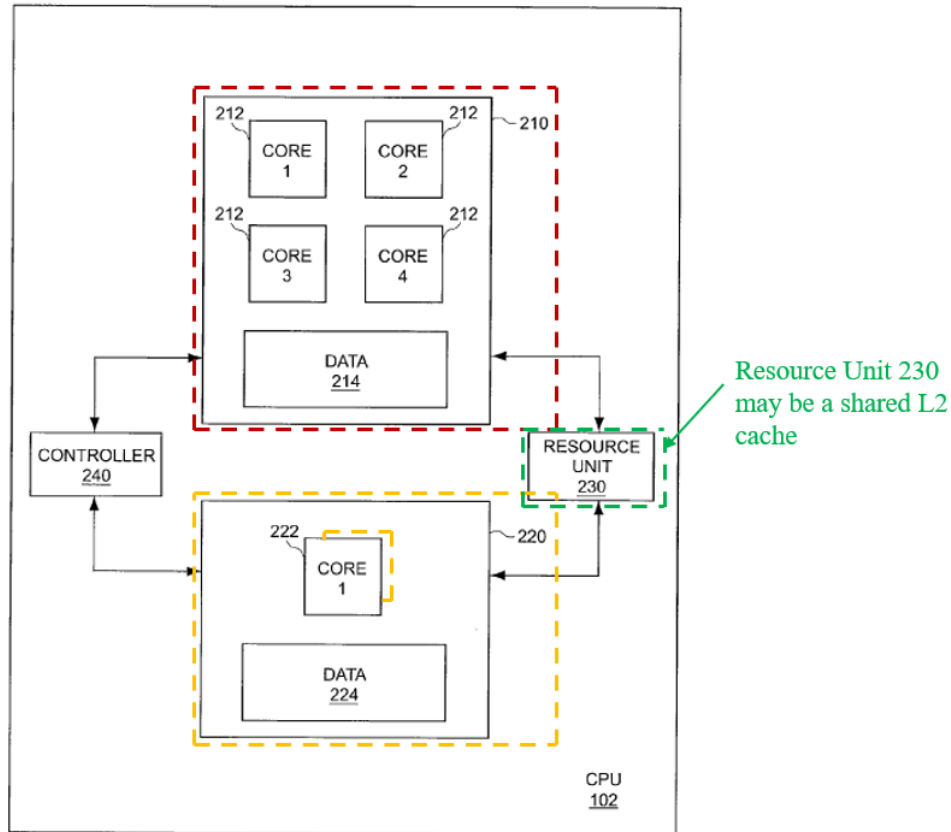
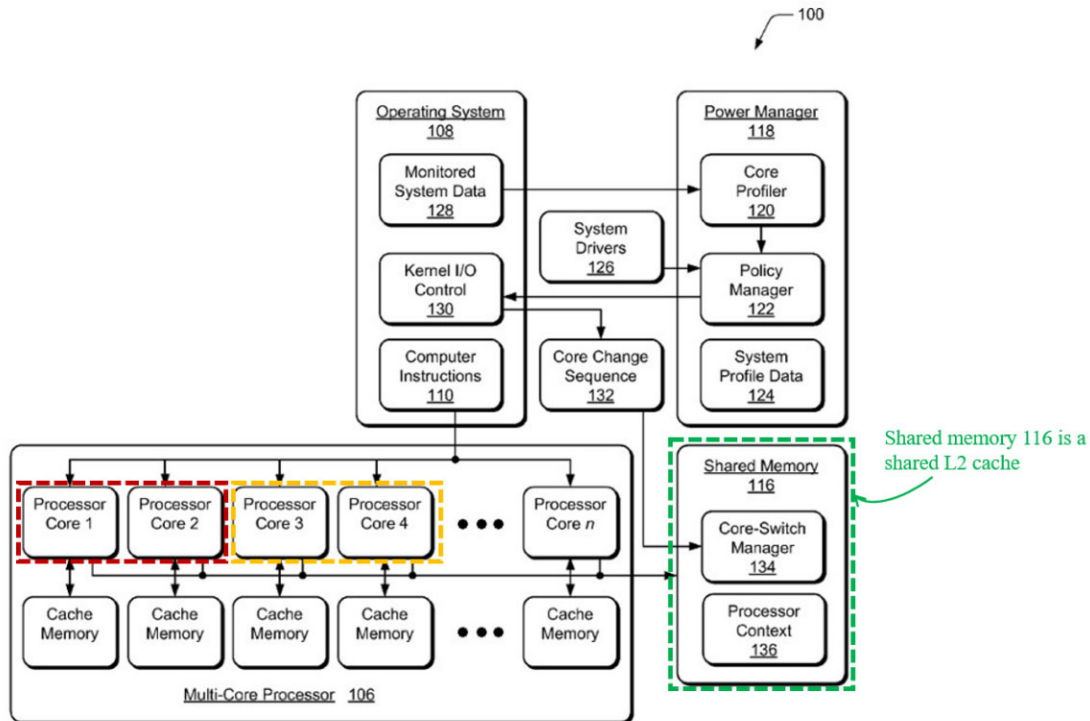


FIG. 2

Id., Fig. 2 (annotated).

168. In the second combination, Sakarda discloses a shared caching layer. For example, Sakarda teaches that a “**shared memory 116** (also referred to as **L2 cache memory** in a multi-core processor system) is accessible to both **processor core 102** and **processor core 104**.” Sakarda, 3:42-45. The figure below, adapted from Sakarda’s Figure 1, shows that **shared memory 116** is accessible to all cores.



Id., Fig. 1 (annotated and altered above).

- f. [1e] “power management hardware to enable and disable the first plurality of cores and the second plurality of cores,”

169. Both Mathieson-Sakarda combinations include “**power management hardware** to enable and disable the **first plurality of cores** and the **second plurality of cores**.”

170. In the first combination, Mathieson’s CPU 102 includes **controller 240**. Mathieson ¶30, Fig. 2. The **controller** “triggers a *hardware mechanism* that stops and powers off the first set of cores 210.” *Id.* ¶43 (emphasis added); *see also id.* ¶3 (“The present invention relates generally to computer hardware and, more specifically, to a system and method for power optimization.”), ¶79 (“[A]spects of

the present invention may be implemented in hardware or software or in a combination of hardware and software.”).

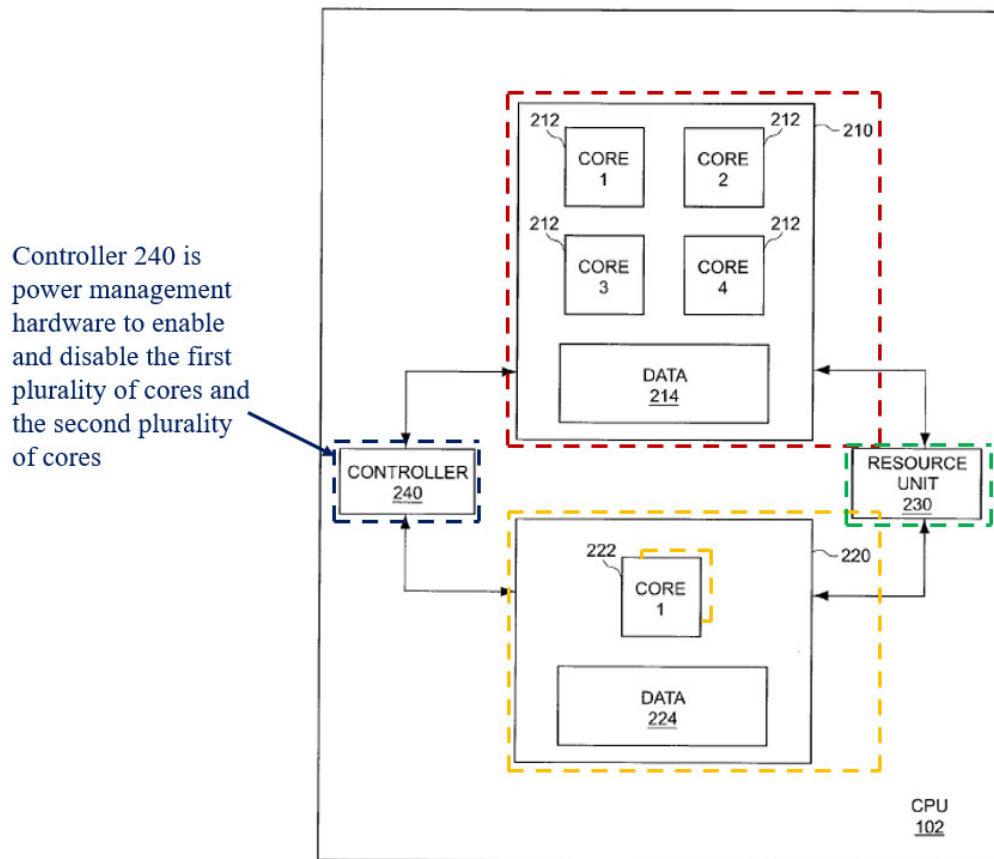


FIG. 2

Id., Fig. 2 (annotated).

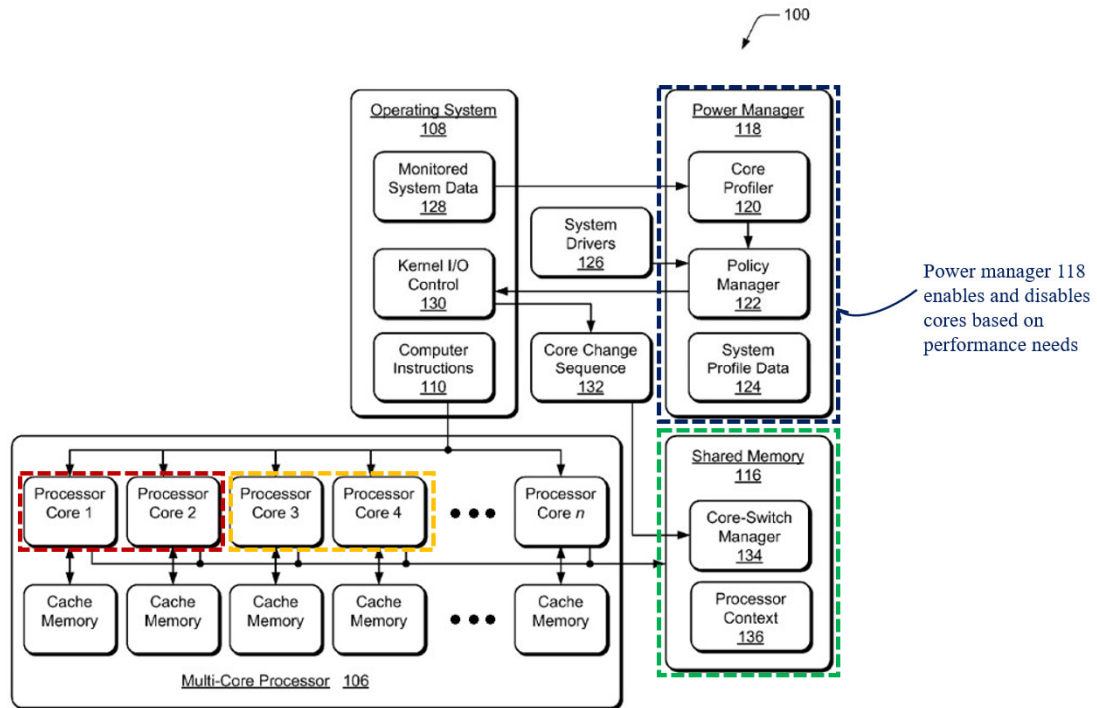
171. **Controller 240** can cause the system to switch operating modes and enable/disable cores. *Id.* ¶30. For example, in a first mode the **first set of cores** is enabled and the **second set of cores** is disabled. *Id.* ¶¶30-31. When the **controller 240** causes CPU 102 to switch to a second mode, it disables the **first set of cores** and enables the **second set of cores**. *Id.*

172. In the second Mathieson-Sakarda combination, Sakarda's **power manager 118** includes hardware to “determine when to switch processor cores to execute the computer instructions 110 based on performance needs and the processing load.” Sakarda, 4:1-4; *see also id.*, 6:50-62. A POSITA would have understood “performance needs” refers to system demand. **Power manager 118** thus enables/disables cores based on demand. Further, **power manager 118** “monitor[s] and implement[s] a system performance and power policy for a multi-core processor system.” Sakarda, 4:34-36. The following table provides an exemplary system performance and power policy.

IPC	DMIPS	Memory Statistics	State of System	Performance Policy		
				Processor Core 1	Processor Core 2	Processor Core 3
IPC < 0.5	0-500	Data Cache Misses > 10%	Memory Bound	Disabled	Disabled	Enabled
IPC < 0.5	500-1000	Data Cache Misses > 10%	Memory Bound	Enabled	Disabled	Disabled
IPC > 0.5	500-1000	Data Cache Misses < 10%	CPU Bound	Enabled	Disabled	Disabled
IPC > 1.0	1000-2000	Data Cache Misses > 10%	Memory and CPU Bound	Enabled	Enabled	Disabled
IPC < 0.5	0-500	Data Cache Misses < 10%	Low MIPS Active and Idle	Disabled	Disabled	Enabled

Id., 4:46-63.

173. The figure below, adapted from Sakarda's Figure 1, illustrates how cores are enabled/disabled based on the system performance and power policy implemented by **power manager 118**.



Id., Fig. 1 (annotated and altered above).

- g. [1f] “wherein an operating system that executes on the multi-core processor is to monitor a demand for the multi-core processor and control a core mix of the first plurality of cores and the second plurality of cores based on the demand with the power management hardware.”

174. Both Mathieson-Sakarda combinations satisfy “wherein an **operating system** that executes on the multi-core processor is to monitor a **demand** for the multi-core processor and control a core mix of the **first plurality of cores** and the **second plurality of cores** based on the **demand** with the **power management hardware**.”

175. In the first combination, Mathieson explains “[t]he transition from the first mode to the second mode, and vice versa, can be done transparently to high-

level software, such as the **operating system**.” Mathieson ¶43. A POSITA thus would have understood an **operating system** executes on Mathieson’s multi-core CPU 102.

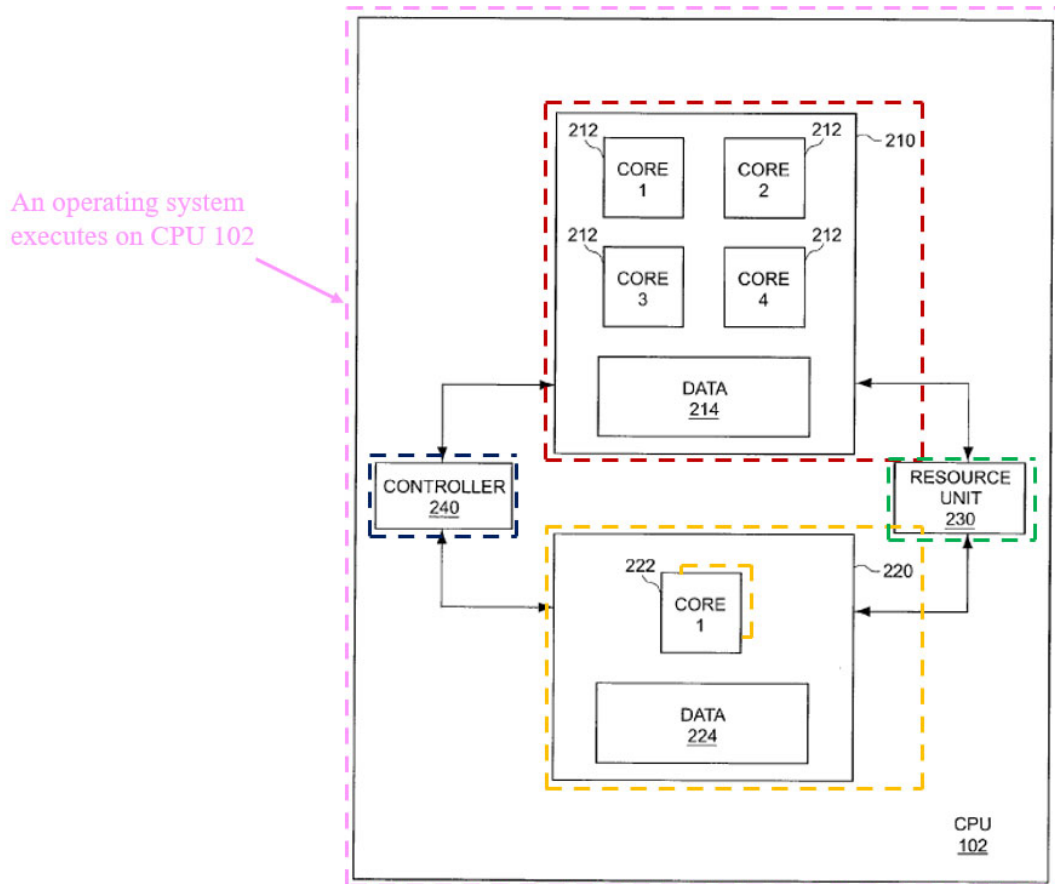


FIG. 2

Mathieson, Fig. 2 (annotated).

176. Mathieson further discloses a **controller 240**, included in CPU 102, that evaluates a **workload**. *Id.* ¶¶30-31. A POSITA would have understood Mathieson’s **workload** refers to **demand**. According to Mathieson, CPU 102 may execute **controller 240**, *see* Mathieson ¶30, which means an **operating system** running on

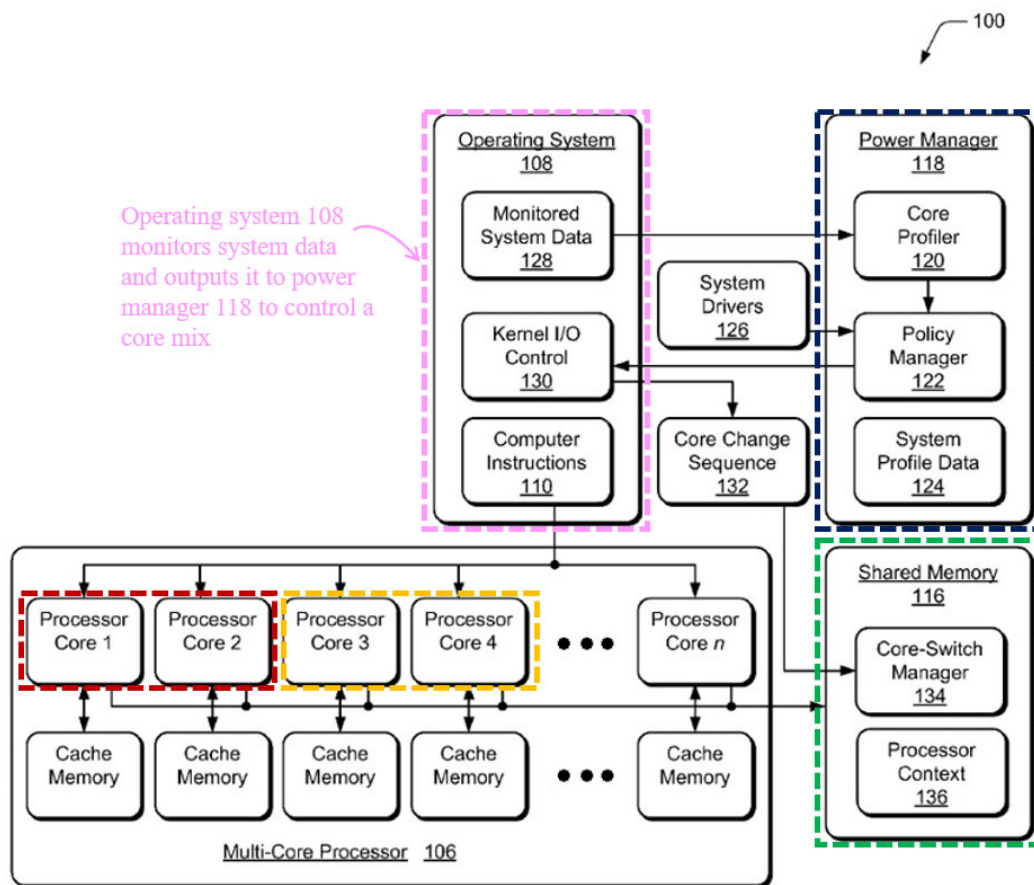
CPU 102 operates **controller 240**,¹ Accordingly, a POSITA would have understood Mathieson's **controller 240** is executed by an **operating system** running on CPU 102.

177. Additionally, Mathieson discloses **power management hardware** that, by enabling/disabling cores or increasing/decreasing their operating frequencies, controls a “core mix” based on demand. *See supra* [1e]; Mathieson ¶¶43, 30-31. Indeed, Mathieson discloses switching between its first mode and second mode, and embodiments in which both sets of cores are operated and “the CPU 102 may operate in both the first mode and the second mode simultaneously,” including during a “transition period” between modes. Mathieson ¶31.

178. In the second combination, Sakarda describes an **operating system 108**. *See* Sakarda, Fig. 1. A “core profiler 120 can receive monitored system data 128 from the **operating system 108**, such as current processing performance requirements for a processing load.” *Id.*, 4:21-24. Sakarda also discloses that power manager 118 and/or core profiler 120 can “monitor and implement a system performance and power policy,” including one in which the system can monitor whether its state is “memory bound” or “CPU bound.” *Id.*, 4:34-64. These system

¹ As a matter of basic knowledge in the art, a computer operating system executes and manages hardware. *See, e.g.*, EX-1016 (“Naghibzadeh”), 1, xiii.

states are known to the **operating system** since the **operating system** knows the memory and CPU utilization. Core profiler 120 is part of **power manager 118**. *Id.*, 3:64, Fig. 1. A POSITA thus would have understood **operating system 108** monitors a demand and outputs it to **power manager 118**. And **power manager 118** controls a core mix based on demand, as described above. *See* [1e]; Sakarda, 4:1-4,4:34-36, 4:46-63. A POSITA thus would have understood that, by outputting demand to **power manager 118**, **operating system 108** controls the core mix using power management hardware.



Id., Fig. 1 (annotated and altered above).

3. **Claim 2, 9, 16: “wherein each of the first plurality of cores and each of the second plurality of cores further include a respective caching layer therein.”**

179. In the first combination, Mathieson explains both the **first set** and the **second set** of cores have **an associated cache**. Mathieson ¶45. Although Mathieson may not explicitly state that each individual core has a **respective caching layer** therein, a POSITA would have found it obvious to include such a caching layer, as was almost universally the case in multi-processor architectures.

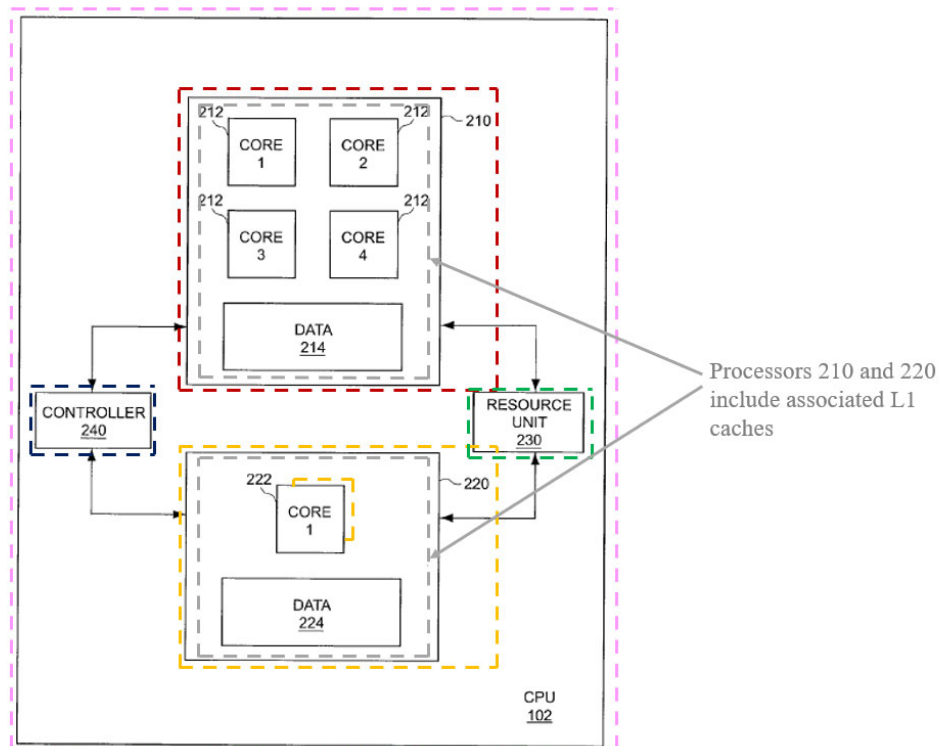


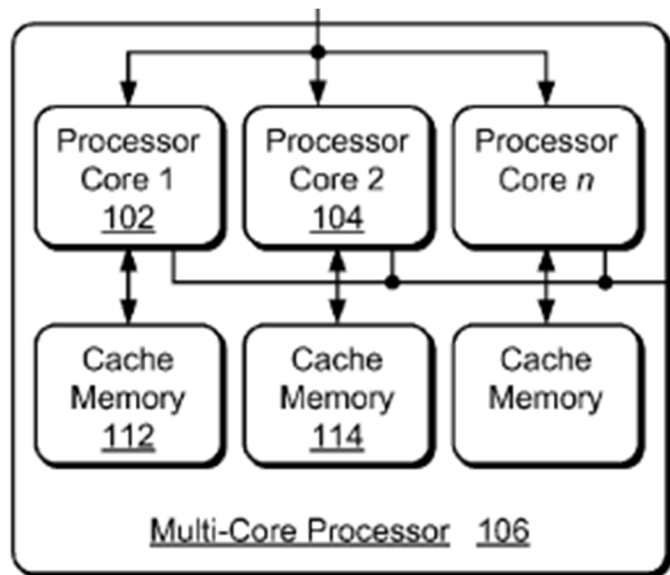
FIG. 2

Id., Fig. 2 (annotated).

180. Mathieson discloses examples in which either the first or second set of cores is only one core. *Id.* ¶28. In those examples, at least one core has its own private

cache. But, more generally, a POSITA would have understood each core should have its own L1 cache, as was overwhelmingly the case in the field of multi-processor architectures at the time (and still today). Sakarda, for example, discloses an **L1 cache** associated with each individual core. Sakarda, 3:38-42. A POSITA would have understood associating a private L1 cache with each core allows faster execution speeds by keeping the relevant data as close as possible to the relevant core, thereby reducing latency. *See* EX-1017 (“Handy”), 90. Because the L1 cache is the fastest and smallest memory level, the L1 cache stores program instructions and data used repeatedly or that the core is likely to need next. *See, e.g.*, Knebel ¶11. Associating an L1 cache with each individual core in Mathieson’s architecture would have improved performance and reduced latencies. That is why associating an L1 cache with each core was so ubiquitous. *See, e.g.*, Patterson-1, 638 (noting such a design “is currently by far the most popular organization”), Figs. 8.1, 8.2, Knebel, Fig. 1, ¶10, EX-1019 (“Lee”), Fig. 1, ¶23. A POSITA would have found it obvious to implement Mathieson in this manner because it would have involved nothing more than combining prior art elements (Mathieson’s L1 caches and the prior art’s teaching (e.g., Sakarda et al.) to use a private L1 cache with each core) according to known methods to yield predictable results. A POSITA would have been motivated to do so to achieve faster operation and lower latencies in Mathieson’s architecture and would have had a reasonable expectation of success in the combination.

181. In the second combination, each of Sakarda's processor cores has "an associated instruction cache and cache memory (also referred to as **L1 cache memory**), such as cache memory 112 that is associated with processor core 102 and cache memory 114 that is associated with processor core 104. Sakarda, 3:38-42, Fig. 1



Id., Fig. 1 (excerpted).

4. **Claims 3, 10, 17: “wherein each of the first plurality of cores includes therein a caching layer for data and a caching layer for instructions.”**

182. A POSITA would have found it obvious to provide each individual core in Mathieson's architecture with its own private L1 cache as Sakarda instructs. *See supra* §VIII.E.3 (Claims 2, 9, 16). Sakarda further teaches that “[e]ach processor core can have **an associated instruction cache and cache memory** (also referred to as L1 cache memory).” Sakarda, 3:38-40. A POSITA would have understood L1

caches are commonly split into instruction data caches. *See, e.g.,* Patterson-1, 382; Sinharoy, 4, Fig. 3; Arimilli, 4:41-45. When Sakarda discloses an associated instruction cache and cache memory, a POSITA would have understood Sakarda's cache memory to be a data cache and not a redundant instruction cache, such that Sakarda's L1 cache is split into an **instruction cache and data cache**.

183. Splitting private L1 caches into separate instruction and data cache layers would have been obvious, because it was textbook. Patterson-1, for instance, teaches that caches may be divided into instruction and data cache layers. Patterson-1, 372, 382. The benefits of doing so include speeding up data access and reducing bottlenecks. *Id.*, 382. Another textbook teaching "The Basics of Caches" teaches that splitting the L1 cache into separate data and instruction cache layers improves the overall cache bandwidth by allowing the core to process both data and instructions on each clock cycle. Patterson-2, 545, 552, 555.

184. A POSITA thus would have been motivated to incorporate a split cache like the one Sakarda describes into Mathieson's architecture to achieve the benefits described by Patterson-1 and Patterson-2. A POSITA further would have recognized that combining Mathieson's architecture with Sakarda's split L1 cache would have involved nothing more than combining prior art elements according to known methods to yield predictable results (e.g., a cache with improved speed/bandwidth

and reduced bottlenecks). A POSITA would have had a reasonable expectation of success in the combination.

185. In the second combination, each of Sakarda's cores has "**an associated instruction cache and cache memory** (also referred to as L1 cache memory)," Sakarda, 3:38-40, and such L1 caches were commonly split into instruction and data caches. *E.g.*, Patterson-1, 382; Sinharoy at 4, Fig. 3. When Sakarda discloses an associated instruction cache and cache memory, a POSITA would have understood the cache memory to be a data cache and not a redundant instruction cache.

5. Claims 4, 11, 18: "wherein the core mix includes a state where some of, but not all of, the first plurality of cores and the second plurality of cores are enabled."

186. Both Mathieson-Sakarda combinations satisfy this feature. In the first combination, Mathieson's **controller 240** controls enablement of the **fast** and **slow** cores, and **power manager 118** controls the switching between fast and slow cores based on demand. *See* [1e], [1f]; Mathieson ¶¶43, 30-31 (disclosing that "the CPU 102 may operate in both the first mode and the second mode simultaneously," including during a "transition period" between modes.).

187. In the second combination, Sakarda discloses a table of states in which at least one of three cores is enabled and at least one core is disabled. Sakarda, 4:36-63. This specific example involves "**two processor cores that have a faster processing rate** and a **third processor core that has a slower processor rate.**" *Id.*,

4:37-40. As explained previously, a POSITA would have found it obvious for Sakarda's system to use multiple cores with a slower processing rate, rather than just one as in this example. *See supra* §VIII.E.1 (“Second Mathieson-Sakarda Combination”), [1a]. In doing so, a POSITA would have found it obvious to include additional system states in which the **slow cores** are operated in a manner analogous to the **fast cores** in the table below, resulting in at least one state where some of, but not all, of the **slow cores** and **fast cores** are enabled.

IPC	DMIPS	Memory Statistics	State of System	Performance Policy		
				Processor Core 1	Processor Core 2	Processor Core 3
IPC < 0.5	0-500	Data Cache Misses > 10%	Memory Bound	Disabled	Disabled	Enabled
IPC < 0.5	500-1000	Data Cache Misses > 10%	Memory Bound	Enabled	Disabled	Disabled
IPC > 0.5	500-1000	Data Cache Misses < 10%	CPU Bound	Enabled	Disabled	Disabled
IPC > 1.0	1000-2000	Data Cache Misses > 10%	Memory and CPU Bound	Enabled	Enabled	Disabled
IPC < 0.5	0-500	Data Cache Misses < 10%	Low MIPS Active and Idle	Disabled	Disabled	Enabled

Sakarda, 4:36-63 (annotated).

6. Claims 6, 13, 20: “wherein the first plurality of cores is at least three.”

188. In the first combination, Mathieson's **first set of cores 210** includes N cores, Mathieson ¶28, which include at least three. Mathieson also discloses embodiments with the **first set of cores 210** having four cores, which is at least three.

Mathieson ¶28, Fig. 2.

189. In the second combination, Sakarda's system includes n cores. Sakarda, 3:3-8, 3:19-21 Fig. 1. A POSITA would have understood the processor could include any number of cores, including any number of **cores—such as three or more—with a faster processing rate**. A POSITA also would have found it obvious to use three or more **fast cores** in Sakarda's architecture based on Mathieson's teachings. Because Sakarda teaches using multiple **fast cores** and Mathieson teaches four **fast cores**, a POSITA would have found it obvious to use three or more **fast cores** in Sakarda's architecture.

190. A POSITA would have understood including multiple fast and slow cores improves system performance by allowing the CPU "to achieve the optimal performance-per-watt ratio" based on how it is operated. Fedorova, 3. A POSITA also would have understood it was common at the time to use three or more fast cores. *See* Sutardja ¶¶219, 248; Mathieson ¶28, Fig. 2. A POSITA thus would have understood using additional fast cores was a known way to increase the capability of a processor to meet higher demands while improving power management of the CPU. Carmack ¶23; EX-1021 ("Variable SMP"), 12 ("[M]ore cores are better for power management" because they can "run at a lower frequency, and hence lower voltage."). At least three fast cores would accommodate increased demands that exceed what two fast cores could accommodate on their own. *See* Sutardja ¶219 (when demand exceeds "that provided by one HP core, multiple HP cores may be

used.”). Using at least three cores thus would have allowed the system to better manage power consumption and process more demanding workloads. Because Sakarda’s architecture is compatible with “multiple processor cores 1, 2, ... n,” Sakarda at 3:3-8, and because using three or more fast cores was common, a POSITA would have reasonably expected to succeed in this arrangement, which would have required mere duplication of high-power cores with no unexpected results. Accordingly, it would have been obvious.

7. Claim 8

a. [8p] “A method comprising:”

191. Mathieson discloses a method. Mathieson, Title, ¶¶3, 8, 54-57, Fig. 4. So does Sakarda. Sakarda, 2:33-35, 7:4-8:3, Fig. 3. Each combination performs such a method.

b. [8a] “monitoring a demand for a multi-core processor by an operating system operating on the multi-core processor,”

192. For the first combination, Mathieson discloses transitioning modes using “high level software, such as the **operating system**” executing on multicore CPU 102, as indicated below. *See* Mathieson ¶43.

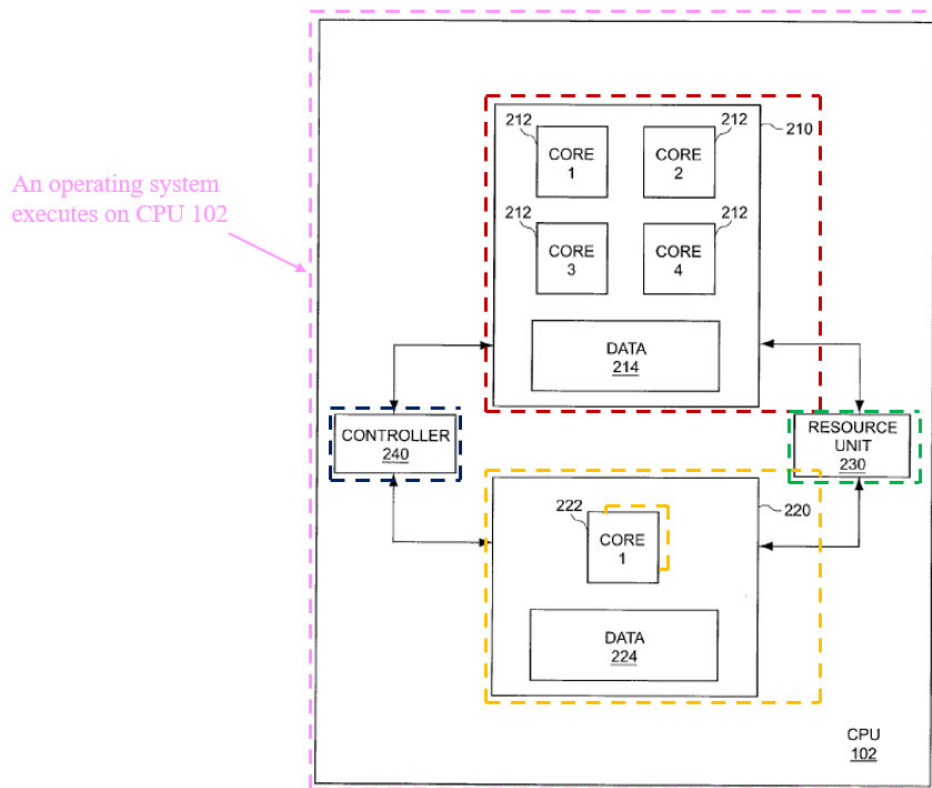


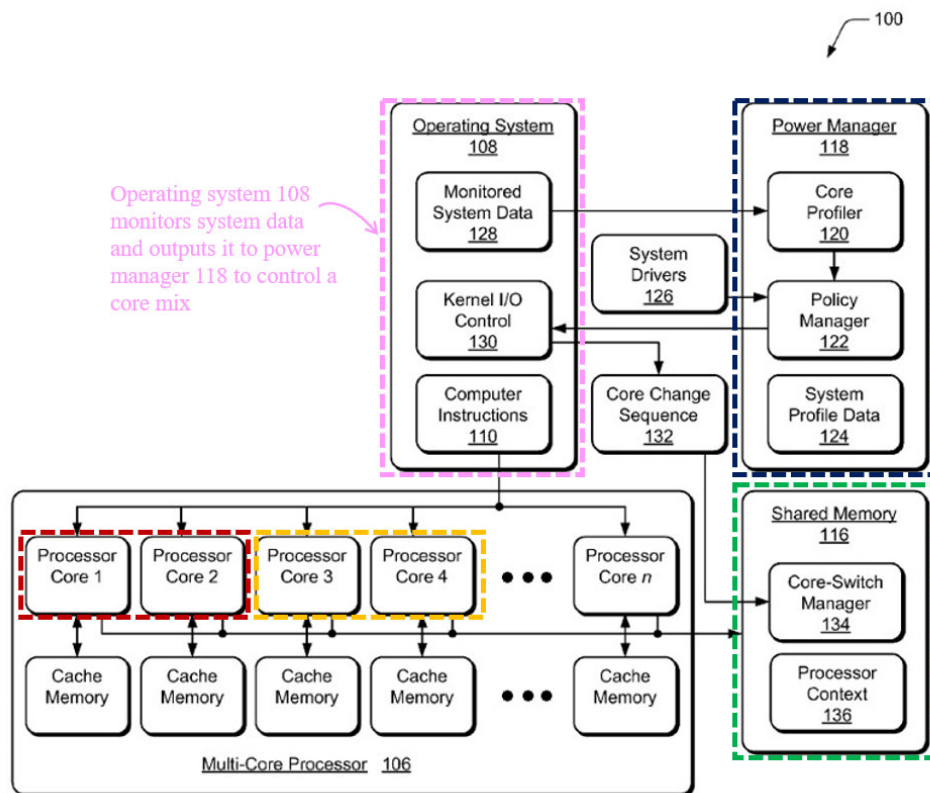
FIG. 2

Id., Fig. 2 (annotated).

193. Mathieson discloses “evaluating ... a workload associated with processing ... one or more operations” to determine whether they should be processed by a second set of cores and, if so, “causing the ... operations to be processed by the second set of cores.” *Id.* ¶8. To do this, **controller 240** evaluates and detects the **workload**, another term for **demand**. *See id.* ¶¶30-31.

194. A computer’s **operating system** executes and manages hardware. Naghibzadeh, 1. When Mathieson explains that CPU 102 executes **controller 240**, the CPU’s **operating system** executes **controller 240**. Mathieson ¶30.

195. For the second combination, Sakarda explains “core profiler 120 can receive monitored system data 128 from the **operating system 108**,” including processing performance requirements for a processing load. Sakarda, 4:21-24. Because core profiler 120 resides in Sakarda’s **power manager 118**, *see id.*, 3:63-4:1, Fig. 1, **operating system 108** monitors demand and outputs it to **power manager 118**.



Sakarda, Fig. 1 (annotated and altered above).

- c. **[8b] “wherein the multi-core processor comprises a first plurality of cores and a second plurality of cores that support a same instruction set,”**

196. Both Mathieson-Sakarda combinations satisfy this feature for the reasons explained in [1a].

- d. **[8c] “the first plurality of cores are higher performance and consume more power than the second plurality of cores,”**

197. Both Mathieson-Sakarda combinations satisfy this feature for the reasons explained in [1b].

- e. **[8d] “each of the second plurality of cores have a maximum operating frequency that is less than a maximum operating frequency of each of the first plurality of cores,”**

198. Both Mathieson-Sakarda combinations satisfy this feature for the reasons explained in [1c].

- f. **[8e] “and a caching layer shared by the first plurality of cores and the second plurality of cores; and”**

199. Both Mathieson-Sakarda combinations satisfy this feature for the reasons explained in [1d].

- g. [8f] “controlling a core mix of the first plurality of cores and the second plurality of cores based on the demand with power management hardware of the multi-core processor.”

200. Both Mathieson-Sakarda combinations satisfy this feature. For the first combination, Mathieson’s **controller 240** switches operating modes by enabling/disabling cores. Mathieson ¶30.

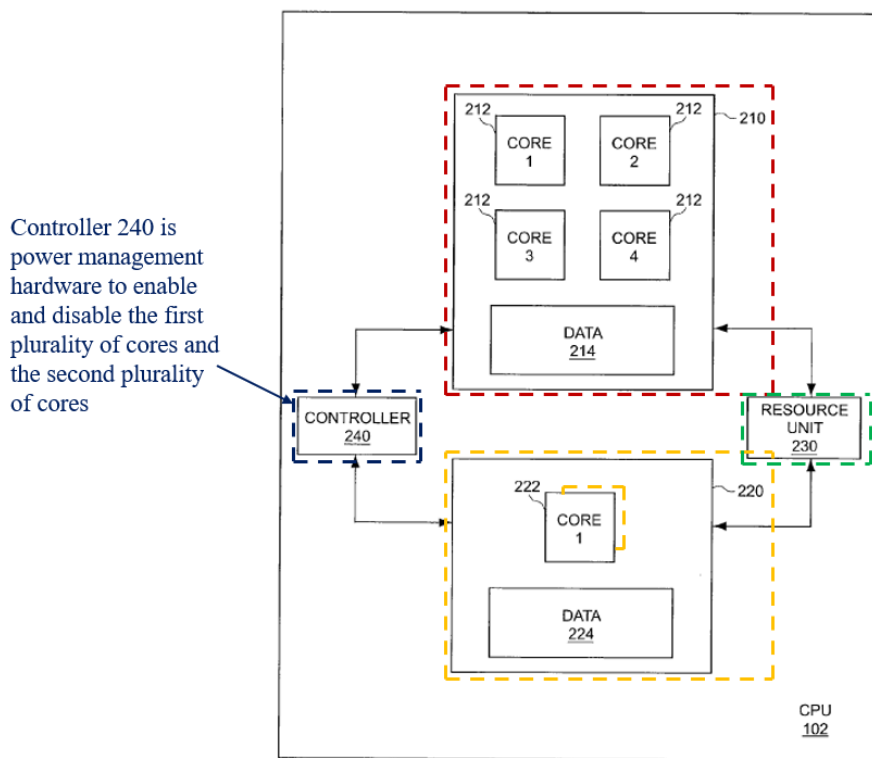


FIG. 2

Id., Fig. 2 (annotated).

201. In a first mode, the **first set of cores** is enabled and the **second set of cores** is disabled. *Id.* ¶¶30-31. When CPU 102 causes **controller 240** to switch to a second mode, the **first set of cores** are disabled and the **second set of cores** are enabled. *Id.* This also includes a “transition period” where “the CPU 102 may

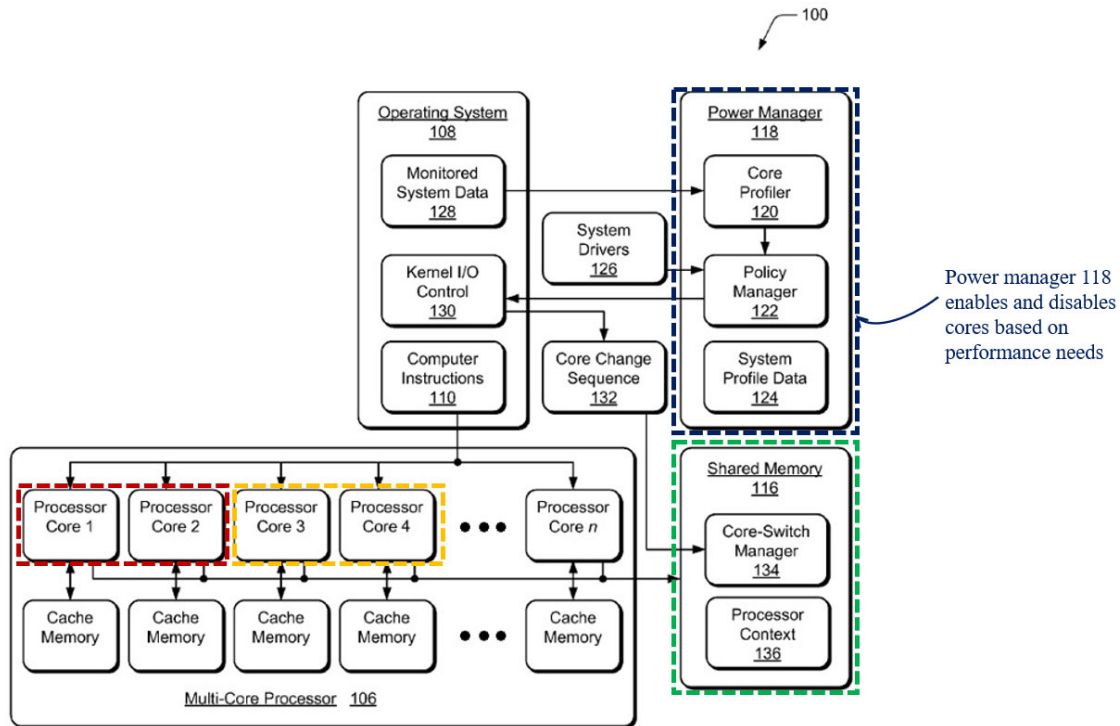
operate in both the first mode and the second mode simultaneously.” *Id.* ¶31. Thus, **controller 240** controls a core mix. **Controller 240** also evaluates and detects a **workload**, synonymous with **demand**. Mathieson ¶¶8, 30-31.

202. For the second combination, Sakarda’s **power manager 118** “determine[s] when to switch processor cores ... based on performance needs and the processing load.” Sakarda, 4:1-4. Here, “performance needs” refers to system demands. Thus, **power manager 118** controls core switching based on demand. The table below provides an exemplary system performance and power policy:

IPC	DMIPS	Memory Statistics	State of System	Performance Policy		
				Processor Core 1	Processor Core 2	Processor Core 3
IPC < 0.5	0-500	Data Cache Misses > 10%	Memory Bound	Disabled	Disabled	Enabled
IPC < 0.5	500-1000	Data Cache Misses > 10%	Memory Bound	Enabled	Disabled	Disabled
IPC > 0.5	500-1000	Data Cache Misses < 10%	CPU Bound	Enabled	Disabled	Disabled
IPC > 1.0	1000-2000	Data Cache Misses > 10%	Memory and CPU Bound	Enabled	Enabled	Disabled
IPC < 0.5	0-500	Data Cache Misses < 10%	Low MIPS Active and Idle	Disabled	Disabled	Enabled

Sakarda, 4:34-64.

203. Different cores are enabled/disabled based on system performance and the power policy implemented by **power manager 118**.



Id., Fig. 1 (amended and annotated).

8. Claim 15

- a. [15p] “A non-transitory machine readable medium containing code that when processed by a machine causes a method to be performed, the method comprising:”

204. For the first combination, Mathieson discloses this feature. *See supra* §VII.G (claim 15). For the second combination, Sakarda’s computer-readable medium 514 provides “types of information and/or data related to operational aspects of device 500.” Sakarda, 9:63-66.

- b. **[15a] “monitoring a demand for a multi-core processor by an operating system executing on the multi-core processor,”**

205. Both Mathieson-Sakarda combinations satisfy this feature for the reasons explained in [8a], [1f].

- c. **[15b] “wherein the multi-core processor comprises a first plurality of cores and a second plurality of cores that support a same instruction set,”**

206. Both Mathieson-Sakarda combinations satisfy this feature for the reasons explained in [8b], [1a].

- d. **[15c] “the first plurality of cores are higher performance and consume more power than the second plurality of cores,”**

207. Both Mathieson-Sakarda combinations satisfy this feature for the reasons explained in [8c], [1b].

- e. **[15d] “each of the second plurality of cores have a maximum operating frequency that is less than a maximum operating frequency of each of the first plurality of cores,”**

208. Both Mathieson-Sakarda combinations satisfy this feature for the reasons explained in [8d], [1c].

- f. **[15e] “and a caching layer shared by the first plurality of cores and the second plurality of cores; and”**

209. Both Mathieson-Sakarda combinations satisfy this feature for the reasons explained in [8e], [1d].

- g. [15f] “controlling a core mix of the first plurality of cores and the second plurality of cores based on the demand with power management hardware of the multi-core processor.”**

210. Both Mathieson-Sakarda combinations satisfy this feature for the reasons explained in [8f], [1f].

F. Ground 2B: Mathieson (EX-1003), Sakarda (EX-1007), and Liu (EX-1006) renders obvious claims 7, 14, and 21

1. Claims 7, 14, 21: “coherency logic circuitry”

211. When multiple cores share memory, as in Mathieson and Sakarda, maintaining cache coherency is fundamental. *See supra* §VIII.D.1 (Ground 1D); Tucker, 19-11; Arimilli, 1:42-46, Akhiani, 1; Censier & Feautrier, Abstract, 1113. Performing this fundamental task requires coherency logic. *See id.* The first and second Mathieson-Sakarda combinations are multiprocessor architectures with shared memory and must maintain cache coherency. *See id.* Cache coherence logic is therefore implicit—and at least obvious—for these architectures to operate.

212. It would have been obvious to use cache coherency logic circuitry. *See supra* §VIII.D.1; Tucker, 19-11; Herrero, 1; Liu ¶¶3-5. A POSITA would have been motivated to apply Liu, for example, to include a snoop filter in Mathieson’s and Sakarda’s shared **L2 caches**. Snoop filters “can reduce the snoop traffic seen by each processor core.” Liu ¶18. A POSITA thus would have applied Liu’s snoop filter in both Mathieson-Sakarda architectures to maintain cache coherency while reducing

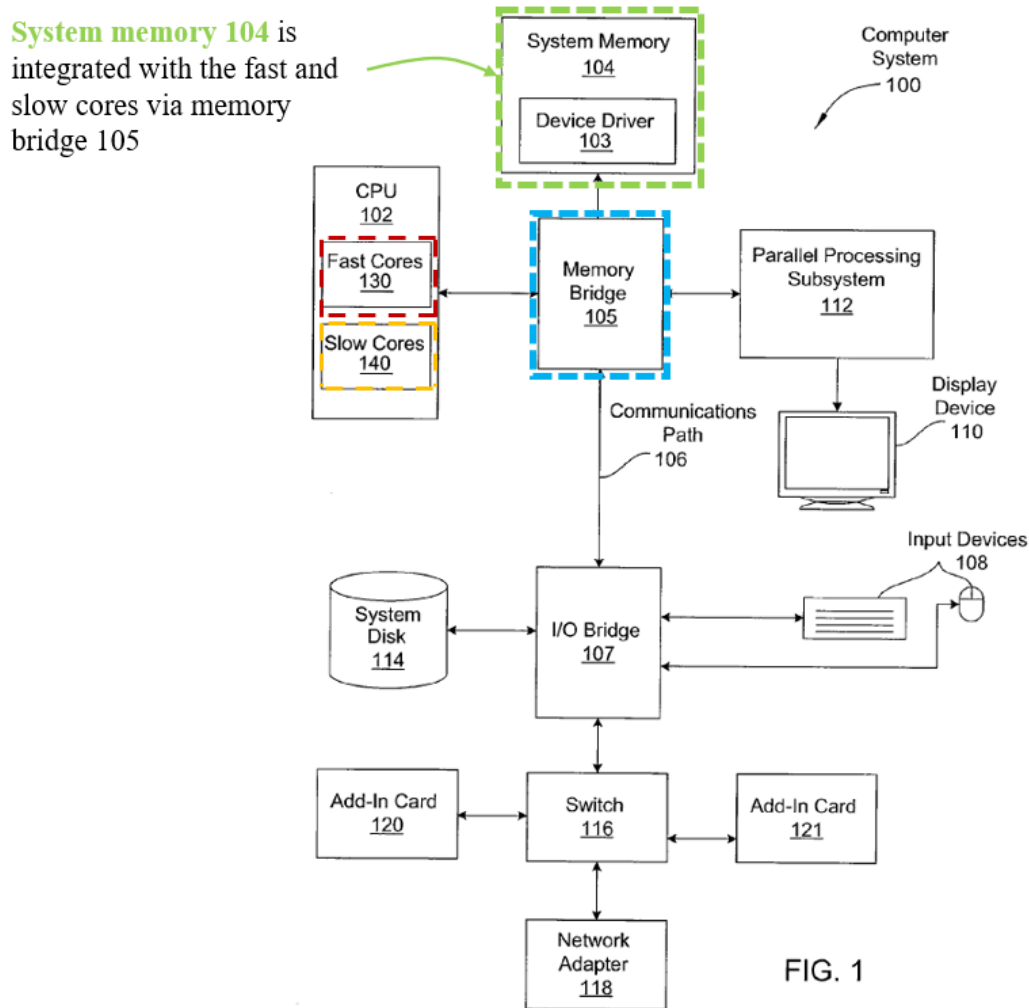
traffic. A snoop filter is “coherency logic circuitry to maintain cache coherency” that would be located in the shared caching layer. *See* Liu ¶4.

213. A POSITA would have recognized both Mathieson-Sakarda-Liu combinations would have involved nothing more than combining prior art elements (coherency logic circuitry, as in Liu, to maintain coherency in both Mathieson-Sakarda caching architectures) according to known methods (snoop coherency circuitry, as in Liu) to yield predictable results (an operational multiprocessor device with reduced traffic to the core caches). A POSITA would have had a reasonable expectation of success in the combination.

G. Ground 2C: Mathieson (EX-1003), Sakarda (EX-1007), and Knebel (EX-1005) renders obvious claims 5, 12, and 19

- 1. Claims 5, 12, 19: “a system memory interface to couple to a system memory to search for a cache line, requested by any core of the first plurality of cores and the second plurality of cores, that is not found in the caching layer” / “coupling a system memory interface of the multi-core processor to a system memory to search for a cache line, requested by any core of the first plurality of cores and the second plurality of cores, that is not found in the caching layer”**

214. A POSITA would have understood that it was conventional and understood that processor systems search a cache and then search memory if the data is not in the cache. However, Mathieson-Sakarda-Knebel include this expressly. In the first combination, Mathieson teaches a **system memory 104** coupled to **fast cores 130** and **slow cores 140** via a **memory bridge 105**. Mathieson, Fig. 1, ¶20.



Id., Fig. 1 (annotated).

215. **Memory bridge 105** “may be integrated into CPU 102” and operates like “a Northbridge chip.” *Id.* ¶20. By this, Mathieson discloses a typical system memory interface. *See id.*

216. Knowing the fundamental operation of caches and cache controllers, a POSITA would have understood Mathieson’s **memory bridge 105** couples **fast cores 130** and **slow cores 140** to **system memory 104** to search for a cache line

requested by a core when the requested cache line is not found in **shared L2 cache 230**. A POSITA would have understood Mathieson's L2 cache controllers (controllers 310 and 320, Mathieson ¶48, Fig. 3) search for a cache line in **L2 cache 230** when a core instructs them to do so. If the cache line requested by the core is not found in **L2 cache 230** (Mathieson's last-layer cache), the requesting core must retrieve the missing cache line from **system memory 104** via **memory bridge 105**. Mathieson ¶20 (CPU 102, which contains the **fast cores 130** and **slow cores 140**, communicates with **system memory 104** via **memory bridge 105**), Fig. 1.

217. A POSITA would have understood that “[w]hen a requested data item is not present in a processor's caches, the item must be fetched from memory.” Tucker at 19-14. That is how caches work. Knebel similarly explains the fundamental notions of a cache hierarchy and cache “hits” and “misses.” Knebel ¶¶10-12. Mathieson's L2 cache controllers work together, like Knebel's cache controller 101, to search the **shared L2 cache** for a cache line requested by one of the cores. *See* Knebel ¶20. If the requested cache line is not in the **shared L2 cache**, it must be retrieved from **system memory 104**. Tucker at 19-14; Knebel ¶12. A POSITA would have understood **memory bridge 105** provides the necessary interface to **system memory 104**. Mathieson ¶20, Fig. 1.

218. Mathieson's **memory bridge 105** is a “system memory interface,” as evidenced by Knebel. A POSITA would have looked to teachings like those in

Knebel because Knebel shares the same cache architecture as the first Mathieson-Sakarda combination (multiple cores, each with private L1 caches, and a shared L2 cache coupled to a system memory). Knebel, Fig. 1. Knebel's cache controller 101 (analogous to Mathieson's L2 cache controllers 310 and 320) searches the cache lines in a shared L2 cache when a core prompts it to do so. Knebel, Fig. 1, ¶¶10-12, 20-22. A POSITA thus would have understood, or at least found it obvious, that Mathieson's L2 cache controllers would operate like Knebel's cache controller 101 (by searching **shared L2 cache** to locate a line having the target address specified by the requesting core).

219. If the cache line requested by one of Mathieson's cores is not in the L2 cache, it would be necessary to search **system memory 104**. When a core requires data stored in a particular address, Knebel's core first determines whether the data is in the lowest level cache (here, L1 cache). Knebel ¶12. If the data is not there, the core next determines whether the data is in the next-level cache (here, the shared **L2 cache**—a last-level cache). *Id.*; *see also* Patterson-2, 545; Censier & Feautrier, 1112. If the data is not found (an event called a cache miss), “the core obtains the data from the **storage 120**.” *Id.*; *accord* Tucker, 19-14. This is textbook. *See, e.g.*, Patterson-2, 550-51.

220. Patterson-2 teaches how to handle cache misses in multi-core processors (like Mathieson's), including “detect[ing] a miss and process[ing] the

miss by fetching the data from memory (or a lower-level cache).”*Id.*, 550. The “basic approach is to stall the CPU” and use a “separate *controller [to] handle[] the cache miss*, [and] fetch[] the data ... from memory.” *Id.*, 551 (emphasis added).

221. A POSITA would have been motivated to apply Knebel’s cache miss techniques to Mathieson’s architecture because, in addition to Knebel’s own teachings, Tucker and Patterson-2 specifically teach fundamental cache miss techniques by using a controller to fetch data from memory (as Knebel teaches). Patterson-2, 550; Tucker, 19-14. Encountering a cache system like Mathieson’s, a POSITA would have applied Knebel’s teachings such that a system memory interface (**memory bridge 105**, which provides the interface to **system memory 104**) would search for the cache line in **system memory 104** in the event of a cache miss. This would have involved no more than combining prior art elements (understanding Mathieson’s L2 cache controllers would apply Knebel’s technique for handling cache misses in systems like Mathieson’s) according to known methods (fetching data from memory after a cache miss) to yield predictable results (a processor that handles cache misses by fetching the data from memory). A POSITA would have had a reasonable expectation of success in the combination.

222. In the second combination, Sakarda discloses **memory 208**, Sakarda, 6:25-28, Fig. 2, which is part of SoC 200 and thus includes system memory. Sakarda, 6:24-28, Fig. 2. SoC 200 “can also include an integrated data bus” that can be

implemented “as a **memory bus or memory controller**.” Sakarda, 6:8-13. **Memory controller 206** thus couples the components of SoC 200, such as **memory 208**, to multi-core processor system 214, including **cores 210** and **cores 212**.²

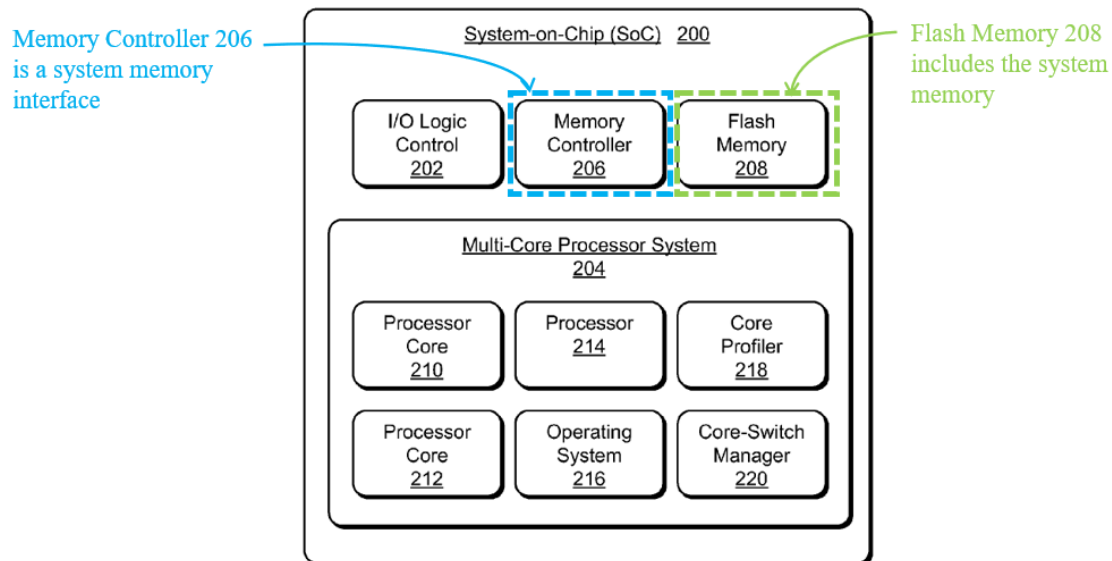


Fig. 2

Sakarda, Fig. 2 (annotated).

223. Because **memory controller 206** couples Sakarda’s processing cores to **memory 208**, it is a system memory interface. *Id.*, 6:8-10. As explained above, a

² **Cores 210** and **cores 212** shown in Figure 2 are the same as **cores 102** and **cores 104** of shown in Figure 1.

POSITA would have understood Sakarda's **memory controller 206** couples **processor core 102** and **processor core 104** to **memory 208** to search for a requested cache line when not found in the **shared L2 cache 116** (Sakarda's last-layer cache). The requesting core would then retrieve the missing cache line from **memory 208** via **memory controller 206**. *See* Sakarda, 6:8-16, 6:24-28; Tucker, 19-14; Knebel ¶¶10-12; Patterson-2, 550-51, 545; Censier & Feautrier, 1112.

224. A POSITA would have applied Knebel's cache miss techniques to Sakarda's architecture based on Knebel's teachings and those of the prior art. *See* Patterson-2, 550; Tucker, 19-14. This would have involved nothing more than combining prior art elements (Knebel's logic for handling cache misses in multiprocessors with similar cache hierarchies, such as Sakarda's architecture) according to known methods (fetching data from memory in the event of a cache miss) to yield predictable results (a processor that handles cache misses by fetching the data from memory). A POSITA would have had a reasonable expectation of success in the combination.

IX. SECONDARY CONSIDERATIONS

225. I have considered the issue of secondary considerations and am currently unaware of any evidence of secondary considerations that would support a finding of non-obviousness or otherwise impact my opinions herein.

X. CONCLUSION

226. For the reasons set forth above, it is my opinion that claims 1-21 of the '281 patent are unpatentable in view of the prior art.

227. In signing this declaration, I understand that the declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I acknowledge that I may be subject to cross-examination in this case and that cross-examination will take place within the United States. If cross-examination is required of me, I will appear for cross-examination within the United States during the time allotted for cross-examination.

228. I declare that all statements made herein of my knowledge are true, that all statements made on information and belief are believed to be true, and that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: Oct 2nd 2024

By:  _____

Murali Annavaram